Human, Motion and Other Priors for Partially-Supervised Recognition

Karteek Alahari

Habilitation à diriger des recherches
Université Grenoble-Alpes

Spécialité : Mathématiques Appliquées et Informatique

Karteek Alahari

Human, Motion and Other Priors for Partially-Supervised Recognition

Soutenue le 28 janvier 2019

Rapporteurs :

Kristen Grauman	Professeur, Univ. Texas at Austin
Vincent Lepetit	Professeur, Univ. Bordeaux
Andrew Zisserman	Professeur, Univ. Oxford

Membres du jury :

Alexei Efros	Professeur, Univ. California at Berkeley
Jean Ponce	Directeur de recherche, Inria
Deva Ramanan	MCF (equiv. HDR), Carnegie Mellon University
Cordelia Schmid	Directrice de recherche, Inria

Summary

This HDR manuscript presents a summary of my research activities after my PhD in 2010. It covers my work in computer vision from the postdoctoral position at Inria Paris to my present researcher position at Inria Grenoble.

Understanding visual data automatically—one of the main challenges in computer vision—is having a significant impact in many practical applications, and this phenomenon can only increase with the continuous rise in digital image and video content being generated. My work presented here focuses on a selection of machine learning methods for computer vision problems. The core theme of these methods is the extraction of priors as additional cues for recognition when only partially-supervised data is available. Such partially-supervised data includes cases where only weak annotations are available, e.g., image or video labels describing the objects in a scene, instead of pixel-wise labels for segmenting objects. It also includes scenarios where data is semi-supervised, e.g., the problem of tracking objects in a video sequence when they are annotated only in the first frame. A third example of partially-supervised data is the case of incremental learning, where an existing model is updated with new training data, in the absence of the original annotations used to train the initial model. In addition to discussing approaches to handle all these scenarios, which lack full annotations, we will also demonstrate the importance of priors for a few fully-supervised recognition problems.

Résumé

Ce manuscrit présente mes activités de recherche en vision artificielle après ma thèse de doctorat soutenue en 2010. Il couvre la période de mon post-doctorat à Inria Paris ainsi que mon activité actuelle de chercheur à Inria Grenoble.

Comprendre automatiquement les données visuelles—un des défis clés en vision artificielle—a un impact significatif dans de nombreuses applications pratiques, et ce phénomène ne fait que s'accentuer avec l'augmentation du contenu généré en images et en vidéos. Mon activité de recherche présentée dans ce manuscript se concentre sur une sélection de méthodes en apprentissage statistique pour résoudre les problèmes en vision artificielle. Le thème central de ces méthodes est l'extraction des a priori en tant qu'informations supplémentaires pour la reconnaissance, lorsque seulement des données partiellement supervisées sont disponibles. Ces données incluent des cas dans lesquels des annotations faibles sont disponibles, par exemple des libellés par image ou vidéo décrivant les objets d'une scène, au lieu de libellés par pixels pour segmenter des objets. Il inclut également des scénarios dans lesquels les données sont semi-supervisées, par exemple le problème du suivi des objets dans une séquence vidéo quand ils sont annotés uniquement dans la première image. Un troisième exemple de données partiellement supervisées est le cas de l'apprentissage incrémental, où un modèle existant est mis à jour avec de nouvelles données d'apprentissage, en l'absence des annotations d'origine utilisées pour former le modèle initial. En plus de discuter des approches pour gérer tous ces scénarios, qui manquent d'annotations complètes, nous allons également démontrer l'importance des a priori pour quelques problèmes de reconnaissance entièrement supervisée.

Contents

Chapter 1

Introduction

The quantity of digital images and videos available on-line continues to grow at a phenomenal speed: home users put their movies on YouTube and their images on Flickr; journalists and scientists set up web pages to disseminate news and research results; and audiovisual archives from TV broadcasts are opening to the public. In 2021, it is expected that nearly 82% of the Internet traffic will be due to videos, and that it would take an individual over 5 million years to watch the amount of video that will cross global IP networks each month by then.[1] Thus, there is a pressing and in fact increasing demand to annotate and index this visual content for home and professional users alike. The available text and audio metadata is typically not sufficient by itself for answering most queries, and visual data must come into play. This manuscript presents methods to address this challenge of computer vision, i.e., understanding visual data.

1.1 Research Contributions

The work presented in this HDR manuscript focuses on a selection of machine learning methods for computer vision problems developed by the author together with his colleagues. In this context, it deals with the aspects of modelling problems, learning the models and their parameters, and performing inference efficiently and accurately. The core theme of the methods discussed in the manuscript is the use of priors as additional cues for recognition when only partially-supervised data is available. Example of such data include: weak image or video labels instead of pixel-wise annotation for segmentation, semi-supervised case for tracking when only the first frame contains object annotation. We will also present the benefit of priors for a few fully-supervised recognition problems, such as text recognition and human pose estimation and segmentation.

The manuscript includes a selection of the author's research contributions during the past eight years. It is organized as three parts: modelling priors, motion understanding, and learning approaches, in the context of visual recognition problems. The following

1. Cisco Visual Networking Index: Forecast and Methodology, 2016-2021, https://goo.gl/C7GQ6s.

presents a short summary of research work conducted between 2003 and 2010, i.e., before and during the author's PhD (Section 1.1.1), and then provides details of the three research axes in the manuscript (in Sections 1.1.2, 1.1.3, 1.1.4).

1.1.1 Research conducted before and during my PhD

My research experience started in the final year of my undergraduate in 2003 at IIIT Hyderabad, India, under the supervision of P. J. Narayanan and C. V. Jawahar. This resulted in a few publications focused on low-level recognition problems: handwritten text in documents [A2], [A6], [A7],[2] geometric prior for object tracking [A5], mixture models for event recognition [A1], [A22].

I then started my PhD in November 2005 under the supervision of Philip H. S. Torr (Professor, Oxford Brookes University during my PhD, and now at University of Oxford). During my PhD, I decided to specialize in combinatorial optimization methods for computer vision—a topic that allowed me to explore research problems in multiple areas, such as machine learning, applied mathematics and computer vision—with a particular focus on labelling problems. I modelled vision tasks defined on images as labelling problems involving several parameters, and developed methods to learn the parameters from annotated training data and perform efficient inference. For example, our work in [A20] extended the subclass of submodular energy functions that can be minimized exactly. It proved that a subclass of functions defining constraints on at most three nodes jointly (i.e., third order functions) can be minimized exactly with the efficient graph cuts algorithm. While this is a new result, it is limited to a rather small set of functions.

To cater to a larger class of energy functions, we proposed efficient approximate algorithms for pairwise functions in [A4], and then extended it to higher order functions [A3], [A26]. Some of the methods in [A4] were shown to be orders of magnitude faster than previous work [90]. Efficient solutions for novel energies, with constraints defined on thousands of image pixels, were introduced in [A14], [A26] (these publications together have over 500 citations[3]). They were targeted at street-level imagery, similar to that in Google Street View, with potential applications in the development of driverless cars and robot navigation. The work in [A14] was also the first one to present an efficient method for solving functions with very high order, and also validate it on the joint task of segmentation and detection.

In summary, the common theme of my PhD work is modelling low-level labelling problems in computer vision as energy functions and solving them efficiently. Since my PhD I have taken this experience to develop approaches: incorporating novel priors into computer vision problems, for motion understanding, and machine learning algorithms. The following three sections will highlight these research contributions after my PhD.

2. References [A#] denote the work of the author, and [#] denote the work of others.
3. All the citation statistics are according to Google Scholar on 10 October 2018.

1.1.2 Modelling priors for visual recognition

Scene text understanding, i.e., recognizing text occurring in images and videos, is one instance where we incorporate priors when formulating the problem in an optimization framework. This has applications such as "reading" text signs on streets in the form of sign boards, street names, in particular when the reader is unable to read, due to unfamiliarity with the language or poor vision. Inspired by work from the natural language processing community, we developed language priors, and used them in combination with visual cues. Previous work addressing this problem relies on localizing characters accurately before forming them into words [181]. Instead of pre-selecting characters sub-optimally to form words, we define a cost function that considers all potential character locations together with natural language priors in the form of pairwise and higher order constraints. This joint model, presented in a series of papers [A16], [A18], [A19], [A21], gave state-of-the-art results on several public benchmark datasets for text recognition. We also applied this formulation to text-based retrieval, wherein image and video content containing text from the given query are searched and retrieved [A17]. Our work on scene text understanding has had a significant impact: (i) over 550 citations in total since 2011, (ii) the datasets we developed as part of this work are the first large-scale ones for scene text problems (with over 1 million images), which were used by other groups in Oxford, Barcelona, Japan, (iii) a PhD thesis award for A. Mishra who worked on this topic.

Our work presented at ICCV 2013 [A8] and in TPAMI 2015 [A23] for segmenting multiple people is motivated by the fact that knowing the locations of body joints (known as body pose) is useful to segment a person, and conversely, person segmentation helps localize body joints more accurately. We proposed priors learnt from estimated body poses to improve the task of person segmentation. We cast this problem in an optimization framework with a novel cost function defined using person detections, and likelihoods computed from pose estimates, colour, motion and stereo disparity cues. This overcame the limitations of several previous works, which considered: (i) segmentation and pose estimation problems separately in spite of the relation between them; or (ii) a single person in the scene; or (iii) people in non-complex poses in uncluttered scenes. Another example where we use human body-specific constraints is to solve the pose estimation problem in videos [A10]. The optimization problem defined on body parts with spatio-temporal links between them is intractable, and previous approaches provide approximate solutions. Although such methods perform well on certain body parts, e.g., head, their performance on lower arms, i.e., elbows and wrists, remains poor. Our approach constrains the location of body parts temporally over video frames to address this issue, but results in a large graph with cyclic constraints. Performing inference on such graphs is a well-known challenging problem in the optimization community. We presented a new approximate scheme based on variable splitting and dynamic programming techniques exploiting pose priors. In doing so, we obtain significant improvement in performance on standard as well as our challenging 'Poses in the Wild' dataset. These works have been popular in the pose estimation and segmentation area, with over 100 citations.

1.1.3 Visual recognition through motion understanding

I started working on different aspects of understanding motion from the beginning of my postdoc career. My first work in this direction was presented at CVPR 2011 [A15], which introduced novel long-range motion and occlusion cues extracted from videos. These cues constrain many pixels in a video temporally to belong to an object, while respecting the relative occlusion ordering between two neighbouring objects. This was the first work to integrate these novel constraints into a video segmentation framework, and extracts object segments missed by previous work. Long-range cues also formed the basis of the work of the first PhD student I co-supervised at Inria Grenoble (Y. Hua). We used them in the context of the tracking problem, where the goal is to localize an object (or several in some cases) in all the frames of a video, given its location in the first frame. More specifically, the cues help estimate the state of an object when tracking it, i.e., whether it is partially or fully occluded, or has undergone a change in appearance or viewpoint [A12]. Further work in this theme of tracking was presented at ICCV 2015 [A13], where we addressed the challenging problem of localizing objects that can undergo transformations, such as severe rotation. The tracking problem in this case is formulated as a proposal selection task, where the location of the object is estimated from a potential set of location proposals. We make two contributions: (i) introducing novel proposals estimated from the geometric transformations undergone by an object and building a rich candidate set for predicting the object location, and (ii) devising a novel selection strategy using multiple cues. This body of work has over 100 citations, and is also the basis of our submission to the VOT-TIR tracking challenge (held at ICCV 2015), where we won one of the competition tracks.

Joint work with P. Tokmakov, another PhD student I co-supervised at Inria Grenoble, first analyzed the fundamental problem of learning motion features automatically from video [A27]. We formulated this as the problem of determining whether an object is in motion, irrespective of camera motion, i.e., independent motion. The core of our approach is a fully convolutional network, which is learned entirely from synthetic video sequences, and their ground-truth optical flow and motion segmentation. This encoder-decoder style architecture first learns a coarse representation of the optical flow field features, and then refines it iteratively to produce motion labels at the original high-resolution. The pixel-level output of this network denotes whether it has undergone independent motion. We then extended this work with a visual memory module to handle cases where the object is in motion only for a part of the video [A28], [A30]. In other words, to segment (an) object(s) that move(s) in at least some part of the video. Given a video frame as input, our approach assigns each pixel an object or background label based on the learned spatio-temporal features as well as the 'visual memory' specific to the video, which is acquired automatically without any manually-annotated frames. The visual memory module is implemented with convolutional gated recurrent units, which allows to propagate spatial information over time. In addition to these publications, we also produced open-source software for this body of work on learning motion features.

1.1.4 Learning approaches for visual recognition

One of my earlier contributions focused on learning the graph structure when finding correspondences between two graphs [A11]. Despite this problem of graph matching being a popular technique in computer vision for finding correspondences between two images, the structure of the graph to perform this matching has been hard-coded in some form up until this work. Our work addressed this important limitation with a simple and effective technique to parameterize a graph model, and learn its attributes from data. It was an oral presentation at ICCV 2013. We proposed to learn a graph model based on a graph representation with histogram-based attributes for nodes and edges. To this end, we presented a generalized formulation for graph matching, which is an extension of previous learning approaches. We show that all attributes of the graph can be learned in a max-margin framework. The proposed method reconstructs a graph model inherent in a target class, and provides impressive matching performance.

The second contribution in this research theme is on weakly-supervised learning for semantic segmentation [A29], where the goal is to assign labels to all the pixels in an image (or video). It is particularly important, as acquiring a training set by labeling images manually at the pixel level is significantly more expensive than assigning class labels at the image level. We present a weakly-supervised learning approach for a fully convolutional neural network (FCNN) based framework, which produces state-of-the-art segmentation results. While there have been recent attempts to learn FCNNs from image-level weak annotations, they need additional constraints, such as the size of an object, to obtain reasonable performance. To address this issue, we present motion-CNN, a novel FCNN framework which incorporates motion cues, and is learned from video-level weak annotations. Our learning scheme to train the network uses motion segments as soft constraints, thereby handling noisy motion information.

Our recent contributions in learning approaches are focused on incremental learning in the context of CNNs [A9], [A25]. Despite the success of CNNs for image classification and object detection, they are ill-equipped for incremental learning, i.e., adapting the original model trained on a set of classes to additionally recognize objects of new classes, in the absence of the initial training data. They suffer from "catastrophic forgetting"–—an abrupt degradation of performance on the original set of classes, when the training objective is adapted to the new classes. Our initial work, presented at ICCV 2017 [A25], addresses this issue, and learns object detectors incrementally, when neither the original training data nor annotations for the original classes in the new training set are available. The core of our proposed solution is a loss function to balance the interplay between predictions on the new classes and a new distillation loss, which minimizes the discrepancy between responses for old classes from the original and the updated networks. This incremental learning can be performed multiple times, for a new set of classes in each step, with a moderate drop in performance compared to the baseline network trained on the ensemble of data. A follow up to this work presented at ECCV 2018 [A9] studies this problem for more general neural network architectures.

1.2 List of Publications

Following is the list of my peer-reviewed publications since PhD (2010), organized according to the publication venue. A subset of these publications, which are included in this manuscript, are also in the bibliography, and are cited in the following chapters.

Refereed journal articles - 5

— P. Tokmakov, C. Schmid, and K. Alahari,
"Learning to Segment Moving Objects," *International Journal on Computer Vision*, 2018 (In press)

— N. Chesneau, K. Alahari, C. Schmid,
"Learning from Web Videos for Event Classification," *IEEE Transactions on Circuits and Systems for Video Technology*, October 2018

— A. Mishra, K. Alahari, and C. V. Jawahar,
"Unsupervised refinement of color and stroke features for text binarization," *International Journal on Document Analysis and Recognition*, June 2017

— A. Mishra, K. Alahari, and C. V. Jawahar,
"Enhancing Energy Minimization Framework for Scene Text Recognition with Top-Down Cues," *Computer Vision and Image Understanding Journal*, April 2016

— G. Seguin, K. Alahari, J. Sivic, and I. Laptev,
"Pose Estimation and Segmentation of Multiple People in Stereoscopic Movies," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, August 2015

Top refereed international conferences (CVPR/ECCV/ICCV) - 17

— F. M. Castro, M. J. Marín-Jiménez, N. Guil, C. Schmid, and K. Alahari,
"End-to-End Incremental Learning," ECCV, 2018

— K. Shmelkov, C. Schmid, and K. Alahari,
"How good is my GAN?" ECCV, 2018

— G. A. Sigurdsson, A. Gupta, C. Schmid, A. Farhadi, and K. Alahari,
"Actor and Observer: Joint Modeling of First and Third-Person Videos," CVPR, 2018

— K. Shmelkov, C. Schmid, and K. Alahari,
"Incremental Learning of Object Detectors without Catastrophic Forgetting," ICCV, 2017

— P. Tokmakov, K. Alahari, and C. Schmid,
"Learning Video Object Segmentation with Visual Memory," ICCV, 2017

— P. Tokmakov, K. Alahari, and C. Schmid,
"Learning Motion Patterns in Videos," CVPR, 2017

— P. Tokmakov, K. Alahari, and C. Schmid,
"Weakly-Supervised Semantic Segmentation using Motion Cues," ECCV, 2016

— Y. Hua, K. Alahari, and C. Schmid,
"Online Object Tracking with Proposal Selection," ICCV, 2015

— Y. Hua, K. Alahari, and C. Schmid,
"Occlusion and Motion Reasoning for Long-term Tracking," ECCV, 2014

— A. Cherian, J. Mairal, K. Alahari, and C. Schmid,
"Mixing Body-Part Sequences for Human Pose Estimation," CVPR, 2014

— K. Alahari, G. Seguin, J. Sivic, and I. Laptev,
"Pose Estimation and Segmentation of People in 3D Movies," ICCV, 2013

— M. Cho, K. Alahari, and J. Ponce,
"Learning Graphs to Match," ICCV, 2013 (**oral**)

— A. Gandhi, K. Alahari, and C. V. Jawahar,
"Decomposing Bag of Words Histograms," ICCV, 2013

— A. Mishra, K. Alahari, and C. V. Jawahar,
"Image Retrieval using Textual Cues," ICCV, 2013

— F. Couzinié-Devy, J. Sun, K. Alahari, and J. Ponce,
"Learning to Estimate and Remove Non-uniform Image Blur," CVPR, 2013

— A. Mishra, K. Alahari, and C. V. Jawahar,
"Top-Down and Bottom-up Cues for Scene Text Recognition," CVPR, 2012

— J. Lezama, K. Alahari, J. Sivic, and I. Laptev,
"Track to the Future: Spatio-temporal Video Segmentation with Long-range Motion Cues," CVPR, 2011

Other refereed international conferences - 4

— U. Roy, A. Mishra, K. Alahari, C. V. Jawahar,
"Scene Text Recognition and Retrieval for Large Lexicons," ACCV, 2014

— V. Goel, A. Mishra, K. Alahari, C. V. Jawahar,
"Whole is Greater than Sum of Parts: Recognizing Scene Text Words," ICDAR, 2013

— A. Mishra, K. Alahari, and C. V. Jawahar,
"An MRF Model for Binarization of Natural Scene Text," ICDAR, 2011 (**oral**)

— M. Schmidt and K. Alahari,
"Generalized Fast Approximate Energy Minimization via Graph Cuts: Alpha-Expansion Beta-Shrink Moves," UAI, 2011

Refereed national conferences - 2

— N. Chesneau, G. Rogez, K. Alahari, and C. Schmid,
"Detecting Parts for Action Localization," BMVC, 2017

— A. Mishra, K. Alahari, and C. V. Jawahar,
"Scene Text Recognition using Higher Order Language Priors," BMVC, 2012 (**oral**)

Chapter 2

Modelling Priors

This chapter is based on the following publications.

— A. Cherian, J. Mairal, K. Alahari, and C. Schmid,
"Mixing Body-Part Sequences for Human Pose Estimation," CVPR, 2014 [PDF]

— A. Mishra, K. Alahari, and C. V. Jawahar,
"Enhancing Energy Minimization Framework for Scene Text Recognition with Top-Down Cues," *Computer Vision and Image Understanding Journal*, April 2016 [PDF]

— G. Seguin, K. Alahari, J. Sivic, and I. Laptev,
"Pose Estimation and Segmentation of Multiple People in Stereoscopic Movies," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, August 2015 [PDF]

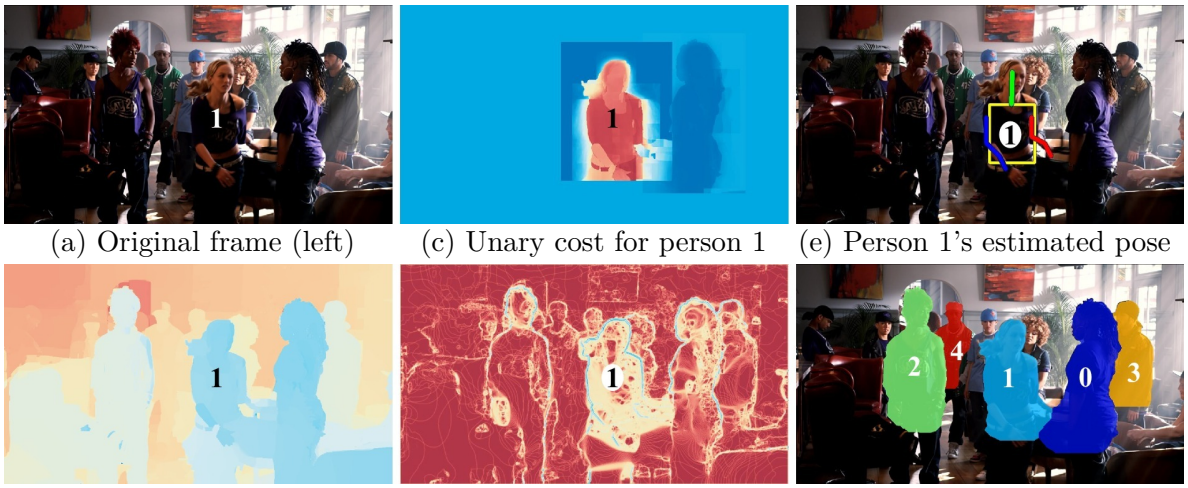
This chapter highlights methods that model priors for computer vision problems when fully-annotated training data is available. In particular, we consider the problems of recognizing scene text in images, multi-person segmentation, and human pose estimation. Our multi-person segmentation method relies on the prior that localizing body joints helps extracting regions corresponding to people (Section 2.1). The second example uses prior information based on human body-specific constraints for the pose estimation problem (Section 2.2). In the case of scene text recognition, we introduce priors inspired by natural language processing models (Section 2.3).

2.1 Multi-Person Segmentation

We explore the problem of detecting and segmenting multiple people in the context of stereoscopic feature length movies, which provide a large amount of readily available video footage of challenging indoor and outdoor dynamic scenes. Our goal is to automatically analyze people in such challenging videos. In particular, we aim to produce a pixel-wise segmentation, and recover the partial occlusions and relative depth ordering of people in each frame, as illustrated in Figure 2.1. Our motivation is three-fold. First and foremost, we wish to develop a mid-level representation of stereoscopic videos suitable for subsequent video understanding tasks such as recognition of actions and interactions of people [198]. Human behaviours are often distinguished only by subtle cues, e.g., a hand contact, and having a detailed and informative representation of the video signal is a useful initial step towards their interpretation. Second, disparity cues available from stereoscopic movies are expected to improve results of person segmentation and pose estimation. Such results, in turn, can be used as a (noisy) supervisory signal for learning person segmentation and pose estimation in monocular videos or still images [2], [63], [121], [195]. For instance, a single 90 minute feature length movie can provide more than 150,000 pixel-wise segmented frames. Finally, pose estimation and segmentation of people will also support interactive annotation, editing, and navigation in stereo videos [91], which are important tasks in post-production and home video applications.

Given the recent success of analyzing people in range data from active sensors, such as Microsoft Kinect [143], [161], and a plethora of methods to estimate pixel-wise depth from stereo pairs [1], the task at hand may appear solved. However, depth estimates from stereo videos are much noisier than range data from active sensors, see Figure 2.1(b) for an example. Furthermore, we aim to solve sequences outside of the restricted "living-room" setup addressed by Kinect. In particular, our videos contain complex indoor and outdoor scenes with multiple people occluding each other, and are captured by a non-stationary camera.

We develop a segmentation model in the context of stereoscopic videos, which addresses challenges such as: (i) handling non-stationary cameras, by incorporating explicit person detections and pose estimates; (ii) the presence of multiple people in complex indoor and outdoor scenarios, by incorporating articulated person-specific segmentation masks (Section 2.1.2) and explicitly modelling occlusion relations among people; and finally (iii) the lack of accurate stereo estimates, by using other cues, such as colour and



(a) Original frame (left) (c) Unary cost for person 1 (e) Person 1's estimated pose

(b) Disparity (d) Smoothness cost (f) Segmentation result

Figure 2.1 – Illustration of the steps of our proposed framework on a sample frame (a) from the movie "StreetDance". We compute the disparity map (b) from the stereo pair. Occlusion-aware unary costs based on disparity and articulated pose mask are computed for all the people detected in the scene. In (c) we show the unary cost for the person labelled 1. Pairwise smoothness costs computed from disparity, motion, and colour features are shown in (d). The range of values in (b,c,d) is denoted by the red (low) - blue (high) spectrum of colours. The estimated articulated pose for person 1 is shown in (e). (f) shows the final segmentation result, where each colour represents a unique person, and the numbers denote the front (0) to back (4) ordering of people. (*Best viewed in colour.*)

motion features. We cast the problem as a discrete labelling task involving multiple person labels, devise a suitable cost function (Section 2.1.1), and optimize it efficiently (Section 2.1.3). We evaluate the proposed model on our new Inria 3DMovie dataset with challenging realistic dynamic scenes from two stereoscopic feature-length movies "StreetDance" [Giwa and Pasquini, 2010] and "Pina" [Wenders, 2011] (Section 2.1.4). Additional comparative evaluations of our method are presented in [A23].

Related work. Some of the previous work is limited to segmenting one or two people [87], or solving detection [81], pose estimation [161], and segmentation [87] problems independently. There have been some attempts to solve these problems jointly, such as joint pose estimation and segmentation [96], [157], but consider restrictive indoor settings [157] or lack an occlusion model to capture interactions among people [96]. The proposed method addresses these issues with an explicit occlusion model for segmenting multiple people in indoor and outdoor scenarios. A more elaborate review of related work is presented in [A23].

2.1.1 The segmentation model

We aim to segment stereoscopic video sequences extracted from 3D movies into regions representing individual people. Figure 2.1 illustrates an overview of our method on a sample frame from a video. Here we consider a stereo pair (only the left image is shown in the figure), estimate the disparity for every pixel, and use it together with person detections, colour and motion features, and pose estimates, to segment individual people, as shown in Figure 2.1(f).

We initialize our model using automatically obtained person detections and assign every detection to a person, i.e., we assume a one-to-one mapping between people and detections. Each pixel i in the video takes a label from the set $\mathcal{L} = \{0, 1, \ldots, L\}$, where $\{0, 1, \ldots, L-1\}$ represents the set of person detections and the label L denotes the "background".[1] The cost of assigning a person (or background) label, from the set $\mathcal{L}$, to every pixel i, $E(\mathbf{x}; \boldsymbol{\Theta}, \tau)$, is given by:

$$E(\mathbf{x}; \boldsymbol{\Theta}, \tau) = \sum_{i \in \mathcal{V}} \phi_i(x_i; \boldsymbol{\Theta}, \tau) + \sum_{(i,j) \in \mathcal{E}} \phi_{ij}(x_i, x_j) + \sum_{(i,k) \in \mathcal{E}^t} \phi^t_{ij}(x_i, x_k), \tag{2.1}$$

which is a special case of the energy (2.16). Here, $\mathcal{V} = \{1, 2, \ldots, N\}$ denotes the set of pixels in the video. The pairwise spatial and temporal neighbourhood relations among pixels are represented by the sets $\mathcal{E}$ and $\mathcal{E}^t$ respectively. The temporal neighbourhood relations are obtained from the motion field [102] computed for every frame. The function $\phi_i(x_i; \boldsymbol{\Theta}, \tau)$ is the cost of a pixel i in $\mathcal{V}$ taking a label x_i in $\mathcal{L}$. It is characterized by pose parameters $\boldsymbol{\Theta} = \{\Theta^0, \Theta^1, \ldots, \Theta^{L-1}\}$ and disparity parameters $\tau = \{\tau^0, \tau^1, \ldots, \tau^{L-1}\}$, where Θ^l and τ^l represent the pose and disparity parameters for a person label l respectively. The disparity parameters determine the front-to-back ordering of people in the scene, as discussed in more detail in Section 2.1.3. Note that the pose and disparity parameters vary across time. However, for brevity, we drop this dependency on t in our notation.

The function $\phi_{ij}(x_i, x_j)$ is a spatial smoothness cost of assigning labels x_i and x_j to two neighbouring pixels i and j. Similarly, $\phi^t_{ij}(x_i, x_k)$ is a temporal smoothness cost. Given the parameters $\boldsymbol{\Theta}$ and τ, minimization of the cost (2.1) to obtain an optimal labelling $\mathbf{x}^* = \arg\min_{\mathbf{x}} E(\mathbf{x}; \boldsymbol{\Theta}, \tau)$, results in segmentation of the video into regions corresponding to distinct people or background. However, in our problem, we also optimize over the set of pose and disparity parameters. In other words, we address the problem of estimating $\mathbf{x}^*$, the optimal segmentation labels, and $\boldsymbol{\Theta}^*$, τ^*, the optimal pose and disparity parameters as:

$$\{\mathbf{x}^*, \boldsymbol{\Theta}^*, \tau^*\} = \arg\min_{\mathbf{x}, \boldsymbol{\Theta}, \tau} E(\mathbf{x}; \boldsymbol{\Theta}, \tau), \tag{2.2}$$

where $E(\mathbf{x}; \boldsymbol{\Theta}, \tau)$ is the cost of label assignment $\mathbf{x}$, given the pose and disparity parameters, as defined in (2.1). Given the difficulty of optimizing E over the joint parameter space, we simplify the problem and first estimate pose parameters $\boldsymbol{\Theta}$ independently of $\mathbf{x}$ and τ as described in Section 2.1.2. Given $\boldsymbol{\Theta}$, we then solve for $\mathbf{x}$, τ as:

$$\{\mathbf{x}^*, \tau^*\} = \arg\min_{\mathbf{x}, \tau} E(\mathbf{x}, \tau; \boldsymbol{\Theta}). \tag{2.3}$$

1. We refer to image regions that correspond to objects other than people as background.

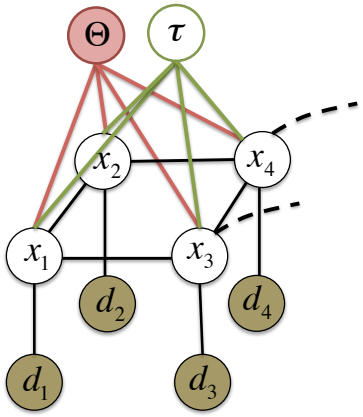


Figure 2.2 – A graphical illustration of our model, where the observed variables are shaded. The variable d_i in the graph represents the features computed at each pixel i in the video. For clarity, we show 4 pixels from a frame, and 2 of the temporal links (dashed line), which connect pixels in one frame to the next. The person label x_i and disparity parameters τ are inferred given the image features d_i, and the pose parameters Θ.

Person 0 Person 1 Person 2 Person 3 Person 4 Background

Figure 2.3 – Illustration of the occlusion-based unary costs for the example in Figure 2.1. From left to right we show the unary costs for persons labelled 0 – 4 and the background. The cost for a pixel to take a label (person or background) is denoted by the red (low) - blue (high) spectrum of colours. Here we observe the effect of accumulating the label likelihoods in a front-to-back order. For example, in the illustration for Person 4, a low cost (red) for taking label 4 is observed only for the pixels that are not occluded by the other people in front. (*Best viewed in colour.*)

Further details on inference are provided in Section 2.1.3. A graphical representation of our model is shown in Figure 2.2. The remainder of this section defines the unary costs, which are computed independently in every frame, and the spatio-temporal pairwise costs in (2.1).

Occlusion-based unary costs. Each pixel i takes one of the person or background labels from the label set $\mathcal{L}$. Building on the approach of [195], we define occlusion-based costs corresponding to these labels, $\phi_i(x_i = l; \mathbf{\Theta}, \tau), l$ in $\mathcal{L}$, as a function of likelihoods β^l, computed for each label l, as follows:

$$\phi_i(x_i = l; \mathbf{\Theta}, \tau) = -\log P(x_i = l|\mathbf{\Theta}, \tau), \tag{2.4}$$

$$\text{where} \quad P(x_i = l|\mathbf{\Theta}, \tau) = \beta_i^l \prod_{\{m|\tau^m > \tau^l\}} (1 - \beta_i^m). \tag{2.5}$$

Here, β_i^l is the likelihood of pixel i taking the person (or background) label l. Note that β_i^l's do not sum to one over the label set for any given pixel. The label likelihood over the entire image β^l is then formed by composing the likelihoods β_i^l, for all pixels $i \in \mathcal{V}$ in the image. In essence, β^l is a soft mask, which captures the likelihood for one person detection. It can be computed using the pose estimate of the person, and image features such as disparity, colour, and motion, as discussed in the following section. To account for the fact that the people in a scene may be occluding each other, we accumulate the label likelihoods in a front-to-back order as in (2.5). This order is determined by the disparity parameters τ we estimate (see Section 2.1.3). In other words, to compute the cost of a pixel taking a person label i, we consider all the other person labels that satisfy $\tau^m > \tau^i$, i.e., are in front of person i. It makes sure that pixel i is likely to take label l, if it has sufficiently strong evidence for label l (i.e., β_i^l is high), and also has low evidence for other labels m, which correspond to people in front of person l (i.e., β_i^m is low for all labels with $\tau^m > \tau^l$). Figure 2.3 shows an illustration of these costs on an example.

Label likelihood β^l. Given a person detection and its corresponding pose estimate Θ^l, the problem of computing the label likelihood β^l can be viewed as that of segmenting an image into person *vs.* background. Note that we do not make a binary decision of assigning pixels to either the person or the background label. This computation is more akin to generating a *soft* likelihood map for each pixel taking a particular person label. We define this using disparity and pose cues as:

$$\beta_i^l = (1 - \alpha^l)\ \psi_p(\Theta^l) + \alpha^l\ \psi_d(\tau^l), \tag{2.6}$$

where $\psi_p(\Theta^l)$ is an articulated pose mask described in Section 2.1.2, $\psi_d(\tau^l)$ is a disparity likelihood, and α^l is a mixing parameter that controls the relative influence of pose and disparity. The disparity potential is given by:

$$\psi_d(d_i; \tau^l, \sigma^l) = \exp\left(-\frac{(d_i - \tau^l)^2}{2(\sigma^l)^2}\right), \tag{2.7}$$

where d_i is the disparity value computed at pixel i. The disparity potential is a Gaussian characterized by mean τ^l and standard deviation σ^l, which together with the pose parameter Θ^l determines the model for person l. We set $\beta_i^L = 0.9$ for all the pixels for the background label L. The method for estimating the parameters τ^l and σ^l for person labels (i.e., for all $l \neq L$) is detailed in Section 2.1.3.

Smoothness cost. In some cases, the disparity cue used for computing the unary costs may not be very strong or may "leak" into the background. We introduce colour and motion features into the cost function (2.1), as part of the smoothness cost, to alleviate such issues. The smoothness cost, $\phi_{ij}(x_i, x_j)$, of assigning labels x_i and x_j to two neighbouring pixels i and j takes the form of a generalized Potts model [20] given by:

$$\phi_{ij}(x_i, x_j) = \begin{cases} \lambda\ (\lambda_1 \exp(\frac{-(d_i - d_j)^2}{2\sigma_c^2}) + \lambda_2 \exp(\frac{-\|\mathbf{v}_i - \mathbf{v}_j\|_2^2}{2\sigma_v^2}) \\ +\lambda_3 \exp(\frac{-(pb_i - pb_j)^2}{2\sigma_p^2})) & \text{if } x_i \neq x_j, \\ 0 & \text{otherwise,} \end{cases} \tag{2.8}$$

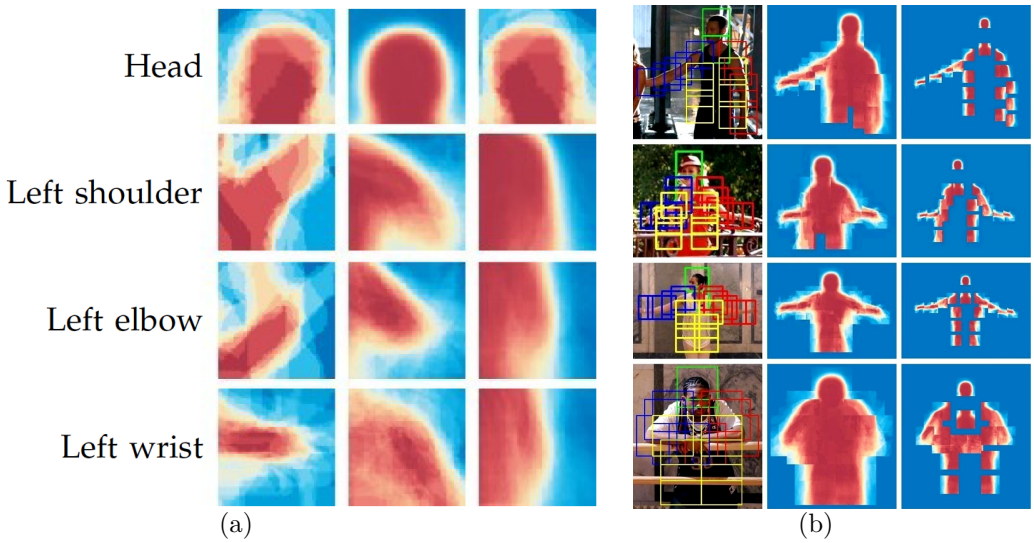


Figure 2.4 – (a) Articulated pose masks for three mixture components are shown for some of the body parts. The pose masks for each part capture a different configuration of the pose. For instance, the masks for "Left wrist" show three different locations of the lower arm: stretched out, partially bent over the shoulder, and lying by the torso. (b) Estimated poses and masks on sample frames. Given a pose estimate (left), we compute a pose-specific mask (middle) using per-mixture part masks learnt from manually segmented training data. We show a scaled version of the masks (right), doubling the actual distances between part masks. This visually explains how each per-mixture mask is contributing to the final mask. The cost for a pixel to take a person label is denoted by the red (low) - blue (high) spectrum of colours in the middle and right figures. (*Best viewed in colour.*)

where λ, λ_1, λ_2, λ_3, σ_c, σ_v and σ_p are parameters of the model. The function $(d_i - d_j)^2$ measures the difference in disparity between pixels i and j. The motion vector at pixel i is denoted by $\mathbf{v}_i \in \mathbb{R}^2$, and $||\mathbf{v}_i - \mathbf{v}_j||_2$ is the ℓ_2-norm of the motion vector difference of pixels i and j. The function $(pb_i - pb_j)^2$ measures the difference of colour features (Pb feature values [9]) of pixels i and j. The temporal smoothness cost $\phi^t_{ij}(x_i, x_k)$ is simply a difference of Pb feature values for two pixels i and k connected temporally by the motion vector $\mathbf{v}_i$.

Thus far we have discussed the model given person detections, their pose and disparity parameters. In what follows, we will describe our method for detecting people, their poses, and the likelihood computed from them (Section 2.1.2). We then provide details of the inference scheme for determining the disparity parameters and the pixel-wise segmentation (Section 2.1.3).

2.1.2 Estimating an articulated pose mask

The aim here is to obtain an articulated pose segmentation mask for each person in the image, which can act as a strong cue to guide the pixel-wise labelling. We wish

to capture the articulation of the human pose as well as the likely shape and width of the individual limbs, torso, and head in the specific pose. We build here on [197], and extend it in two directions. First, we incorporate disparity as input to take advantage of the available stereo signal. Second, we augment the output to provide an articulated pose-specific soft-segmentation mask learnt from manually annotated training data.

We first detect people with a variant of the deformable part-based person detector [49] adapted to appearance and disparity features, and then fill-in any missing detections by interpolating the location of the bounding box. This also smooths the detections temporally [44]. More details on this are available in [A23]. We estimate the pose of the person within each person detection bounding box building on [197]. We restrict our pose estimation models to upper body poses, which are more commonly found in movie data. Again, to benefit from the stereo video, we extract both appearance and disparity features in the frame. In this framework, the model is represented as a set of K parts, where a part refers to a patch centered on a joint or on an interpolated point on a line connecting two joints. For example, we have one part for an elbow, one for a wrist, and two parts between the elbow and the wrist, spread uniformly along the arm length.

Articulated pose mask ψ_p. The output of the pose estimator is the location of the individual parts in the frame as shown in the left column of Figure 2.4(b). To obtain a pose-specific mask we learn an average mask for each mixture component for each part. This is achieved by applying the trained pose-estimator on a training set of people with manually provided pixel-wise segmentations. An illustration of masks for individual parts and mixture components is shown in Figure 2.4(a).

At test time, given an estimated pose with an instantiated mixture component c^* for a part k, the likelihood for the person, $\psi_p(\Theta, i)$ at pixel i, is obtained by laying out and composing the articulated masks m_{kc^*} for all the parts. An illustration of the articulated pose masks for various examples is shown in Figure 2.4(b).

2.1.3 Inference

Poses Θ^l are estimated independently for each person l and fixed throughout the rest of the inference procedure described next. The aim is to compute the optimal disparity parameters τ^* and pixel labels $\mathbf{x}^*$ given the pose parameters $\mathbf{\Theta}$, as described by the minimization problem (2.3). It is well known that minimizing multi-label functions such as $E(\mathbf{x}; \mathbf{\Theta}, \tau)$, which corresponds to the segmentation problem, given the pose and disparity parameters, is in itself NP-hard (for the type of smoothness cost we use) [19]. The additional complexity of optimizing over disparity parameters τ further adds to the challenge. We propose a two-step strategy, where we first: (i) estimate the optimal disparity parameters τ^* using an approximation to (2.3), without the pairwise terms; and then (ii) obtain the pixel labels $\mathbf{x}^*$ with the estimated (and now fixed) parameters τ^* by minimizing the full cost (2.1), as described below.

Obtaining disparity parameters. The estimation of the set of disparity parameters τ for all the people in a frame can be succinctly written as:

$$\tau^* = \arg\min_{\{\tau\}} \tilde{E}(\tilde{\mathbf{x}}; \boldsymbol{\Theta}, \tau), \tag{2.9}$$

where we approximate the original cost function (2.1) by only using unary and ignoring the pairwise terms[2] as $\tilde{E}(\mathbf{x}; \boldsymbol{\Theta}, \tau) = \sum_{i \in \mathcal{V}} \phi_i(x_i; \boldsymbol{\Theta}, \tau)$. Note that for this modified cost function, the optimal pixel labelling $\tilde{\mathbf{x}}$ for a given τ can be obtained independently for each pixel as $\tilde{x}_i = \arg\min_{m \in \mathcal{L}} \tilde{E}(x_i = m, \boldsymbol{\Theta}, \tau)$. Further, the disparity parameter τ is inversely related to depth, and determines the front-to-back order of people in a frame. Thus, this minimization problem (2.9) explores various combinations of the relative order of people in a frame by optimizing over $\{\tau\}$. The set of possible disparity parameter values for each person can still be large, and exploring the exponentially many combinations for all the people in the frame may not be feasible. To address this issue, we obtain and optimize over a small set of (up to 3) candidates $\{\tau^l\}$, for each person l. Using a thresholded pose mask, we compute mean disparity μ^l of all the pixels within, and set $\{\tau^l\} = \{\mu^l, \mu^l \pm \sigma^l\}$. The parameter σ^l is set according to a linear decreasing function of μ^l. Note that the disparity parameters are estimated jointly for all the people in the scene.

Person segmentation. With the estimated disparity (and pose) parameters, we compute the unary and smoothness costs, and use the efficient α-expansion algorithm [21] to optimize (2.1). This assigns every pixel a person or background label from the set $\mathcal{L}$.

2.1.4 Results: Segmenting multiple people

In this manuscript we focus on the segmentation results. Details of the Inria 3DMovie dataset and further analysis of the method is presented in [A23]. In the quantitative evaluation of the segmentation model, shown in Table 2.1, using ground truth annotations, we compare three variants of our approach and two baseline methods. The first one ("No mask, single frame") refers to the case where the label likelihood $\beta_i^l = \psi_d$, i.e., there is no influence of pose on the segmentation. In other words, this method uses disparity features, but not the pose information. The second method ("Uni mask, single frame") incorporates a person location likelihood, which is computed by averaging ground truth segmentations of people from the training data (after rescaling them to a standard size) into a single non-articulated "universal" person mask – an approach inspired by the successful use of such masks in the past [195]. We use this as the *person* likelihood ψ_p, and combine it with disparity likelihood ψ_d, as explained in Section 2.1.1. The third variant ("Pose mask, single frame") incorporates the articulated pose mask, described in Section 2.1.2. Our complete model ("Proposed") introduces temporal smoothness across frames.

For the "Colour only" baseline, we used a colour-based model for the unary costs without the disparity potential. These costs were computed from colour histograms for

2. We note that this is a reasonable approximation, as τ only directly affects the unary cost ϕ_i in (2.1).

Method	Precision	Recall	Int. vs Union
Proposed	**0.869**	**0.915**	**0.804**
Variants of our method:			
No mask, single frame	0.525	0.371	0.278
Uni mask, single frame	0.783	0.641	0.544
Pose mask, single frame	0.849	0.905	0.779
Baselines:			
Colour only	0.778	0.769	0.630
[39]	0.762	0.853	0.662

Table 2.1 – Evaluation of pixel-wise person segmentation on our Inria 3DMovie dataset. We used precision, recall and intersection *vs.* union scores to compare the methods. Our method (*"Proposed"*), which uses disparity, colour, and motion features, along with pose likelihoods and temporal terms shows the best performance. We also show results of variants of our approach and two baseline methods.

each label [20]. In other words, each label is associated with a histogram computed from a region in the image, and the unary cost of a pixel is a function of the likelihood of the pixel, given its colour, taking this label. The success of this model certainly depends on the regions used for computing the histograms. We used the result obtained by segmenting in the disparity space, i.e., "No mask, single frame", as these regions. We believe that this provides a reasonable estimate for the label potentials. The background histogram was computed with bounding boxes harvested from regions with no person detections. Another baseline we compared with, is derived from the recent work of [39], which computes the pose of a person in a scene. We evaluated the (monocular) person *vs.* background segmentation performed as part of this formulation on our dataset.

We used the precision, recall, and intersection *vs.* union [2] measures to evaluate our segmentation results. From Table 2.1, our method "Proposed" shows the best performance. The poor performance of the "Colour only" method, despite a reasonable initialization for the histograms, is perhaps an indication of the difficulty of our dataset. From Figures 2.1 and 2.5 we note that the person *vs.* background distinction is not very marked in the colour feature space. Furthermore, these images appear to be captured under challenging lighting conditions.

We then evaluated the benefits of the temporal smoothness terms in (2.1). Performing segmentation temporally shows a 2% increase in the intersection *vs.* union score (Table 2.1). We also observe that it reduces flickering artifacts, produces more consistent segments and reduces leaking in the segmentation. Results on a few sample frames for the "Proposed" method are shown in Figure 2.5. The success of our approach depends on the quality of detections. Here, we operated in the high-precision mode, at the expense of missing difficult examples, e.g., heavily occluded people. Other prominent failure modes of our method are: (i) challenging poses, which are very different from the training data; and (ii) cases where the disparity signal is noisy for people far away from the camera (e.g., Figure 2.5, row 1).

(a) Original image (b) Segmentation result

Figure 2.5 – Qualitative results on images from the movies "StreetDance" and "Pina". Each row shows the original image and the corresponding segmentation. Rows 1 and 2 demonstrate successful handling of occlusion between several people. The method can also handle non-trivial poses, as shown by Rows 3 and 4. The segmentation results are generally accurate, although some inaccuracies still remain on very difficult examples. For instance, in Row 1, the segmentation for the people in the background for persons 3 and 5, due to the weak disparity cue for these people far away from the camera. The numbers denote the front (low values) to back (high values) ordering of people. (*Best viewed in colour.*)

2.1.5 Summary

The model presented here explicitly represents occlusions, incorporates person detections, pose estimates, and recovers the relative depth ordering of people in a scene. Results suggest that disparity estimates from stereo video, while noisy, can serve as a strong cue for localizing and segmenting people. They also demonstrate that a person's pose, incorporated in the form of an articulated pose mask, provides a strong shape prior for segmentation. The developed representation presents a building block for modelling and recognition of human actions and interactions in 3D movies.

2.2 Human Pose Estimation

The task of estimating human body-joint locations (i.e., human pose) is challenging due to several factors, such as the diversity of appearances, changes in scene illumination and camera viewpoint, background clutter, and occlusion. The difficulty is also not uniform across body parts, e.g., the estimating the location of head is easier than localizing parts corresponding to lower arms, i.e., elbows and wrists. The focus of this work is to improve human pose estimation, and in particular to localize lower-arm parts accurately by leveraging priors on human body parts.

A popular representation for articulated human poses is a set of rigid body parts [7], [39], [50], [53], [197], for which body-part templates are learned from training data. A probabilistic graphical model, often a Markov random field (MRF), is designed with scores provided by these templates. For single images, the MRF is usually modeled as a tree or a star-shaped graph, leading to tractable and efficient inference, as successfully done in [39], [50], [197]. One way to extend such methods for estimating poses in videos is by introducing regularization on the body parts across time, e.g., by adding temporal part-part edges [52], [154], [166], [180], [185]. The resulting graphical model is no longer a tree, and inference becomes intractable. Thus, approximations are required, which can be done by changing the graph structure, e.g., ensemble of tree-structured MRFs [154], [185], or by using approximate inference methods, such as loopy belief propagation or sampling [52], [166], [180].

We present a new approximation scheme adapted to the human pose estimation problem. We begin by generating a set of pose candidates in each frame with a model including temporal links with subsequent frames for the less-certain parts, namely elbows and wrists, see Figure 2.6b. Since the loops in the corresponding MRF are isolated, we show that inference can be performed efficiently with the use of distance transforms [51]. We then compute the n-best poses [13], [128] in each frame to obtain a diverse set of candidate poses (Section 2.2.3). Next we introduce an effective method to smooth these poses temporally. We decompose the human pose into limbs and track them to generate body-part sequences. We then recompose the complete pose by mixing these part sequences (Figure 2.6c, Section 2.2.4). This procedure explores a set of poses that is exponential in K, the size of the candidate set, in polynomial time ($\mathcal{O}(NTK^2)$, where N is the number of body parts and T is the number of frames).

(a) (b) (c)

Figure 2.6 – Human pose estimated by (a) Yang & Ramanan's method [196], our approach: (b) local temporal model for less-certain parts, and (c) mixing body-part sequences.

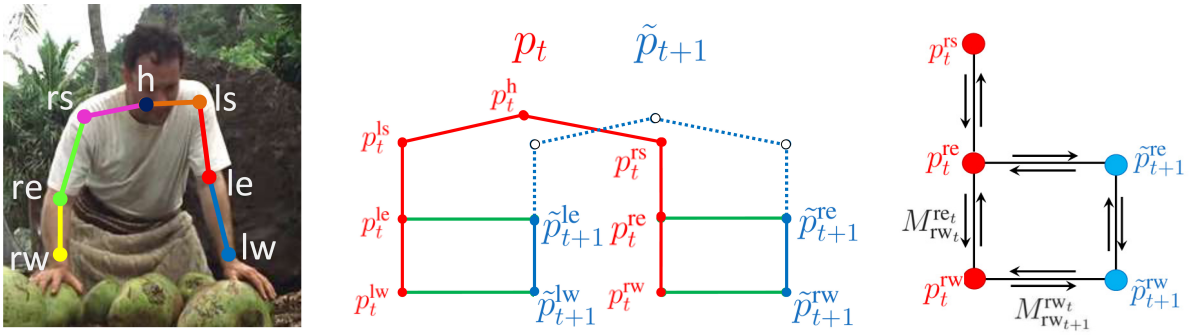
Related work. One of the main challenges for pose estimation in videos is to handle the temporal coupling of parts across frames. This results in models that are highly inter-connected (i.e., loopy graphs with high tree-width) and are thus intractable to perform inference on. Previous works have resorted to approximate inference to address this issue. For instance, the methods in [166], [180] use a sampling approach. Methods such as [52], [98] have used loopy belief propagation instead. They typically come with high computational costs for marginal improvements in performance. Others, e.g., [154], approximate the model as a convex combination of tree-structured graphs linked with dual variables, and solve it with a dual decomposition algorithm. A popular alternative to these approximations is to detect the pose in a few frames, and track it in the rest of the sequence [163], [167], similar to a tracking-by-detection scheme [6], [13], [139]. Further discussion on related work is available in [A10].

2.2.1 Pose estimation in single images

Our work relies on the deformable mixture-of-parts model proposed for single images in [196] due to its performance and computational efficiency. We first briefly present this technique in this section, and then introduce our approach for video sequences. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote a graph with vertices $\mathcal{V}$ and edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ representing the structure of a human pose. Each vertex corresponds to a body part (i.e., head, shoulders, elbows, wrists), and each edge represents a connection between two of these parts; see Figure 2.7a. We define a pose p with respect to this graph $\mathcal{G}$ as a set of 2D coordinates representing the positions of the different body parts in an image as:

$$p = \left\{ p^u = (x^u, y^u) \in \mathbb{R}^2 : \forall u \in \mathcal{V} \right\}.$$

The formulation of [196] uses a mixture of body part models. Every part can be associated with one of M possible "types", and choosing a type configuration determines the pose



(a) (b) (c)

Figure 2.7 – (a) Our graphical model for human pose in a single image is shown with the body parts (**h**ead, **l**eft and **r**ight **s**houlders, **e**lbows and **w**rists) and their spatial connections. (b) The graphical model used to refine the pose estimate p_t in image I_t with a dummy pose $\tilde{p}_{t+1}$ (shown in solid blue lines), which contains only the wrist and the elbow parts in image I_{t+1}. The temporal links between these two poses are shown in green. (c) An illustration showing messages between some of the body parts.

model. Thus, estimating the pose involves not only choosing the part positions p, but also the type for each body part [196]. We use this framework in the paper, but omit the details in the following presentation to simplify the notation.

The single-image pose estimation problem is then formulated as the minimization of the following *cost* $C(I, p)$ for a pose p and an image I:

$$C(I,p) := \sum_{u \in \mathcal{V}} \phi_u(I, p^u) + \sum_{(u,v) \in \mathcal{E}} \psi_{u,v}(p^u - p^v), \tag{2.10}$$

where $\phi_u(I, p^u)$ is an appearance term for the body part u at the position p^u in I, and $\psi_{u,v}(p^u - p^v)$ is a deformation cost for body parts (u, v), which is often compared to the energy model of a spring. Both ϕ_u and $\psi_{u,v}$ have underlying linear filters that are learned by using a structured SVM formulation. When $\mathcal{G}$ is a tree, the exact minimizer of (2.10) can be found in polynomial time with dynamic programming [196].

2.2.2 Pose estimation in videos

Given a video sequence $\mathcal{I} = (I_1, I_2, \ldots, I_T)$, it is common to introduce temporal links between every pair of frames I_t and I_{t+1} in the sequence, in order to impose temporal consistency in the estimation of the pose positions $p_1, p_2, \ldots, p_T$. This is achieved by adding a temporal edge between every pair of nodes p_t^u and p_{t+1}^u, leading to the following cost function:

$$C(I_T, p_T) + \sum_{t=1}^{T-1} C(I_t, p_t) + \lambda_1 \theta(p_t, p_{t+1}, I_t, I_{t+1}), \tag{2.11}$$

where θ is a consistency term between the poses in two consecutive frames and λ_1 is a regularization parameter. We measure the consistency between p_t and p_{t+1} by compar-

ing p_{t+1} with p_t adjusted with optical flow as follows:

$$\theta(p_t, p_{t+1}, I_t, I_{t+1}) = \sum_{u \in \mathcal{V}} \|p^u_{t+1} - p^u_t - f_t(p^u_t)\|^2_2, \tag{2.12}$$

where $f_t(p^u_t)$ is the optical flow between frames I_t and I_{t+1} evaluated at the position p^u_t. Indeed, this approach is quite natural and similar formulations have been proposed [13], [128], [154]. Our work mainly differs from these approaches in the way we address the problem of minimizing (2.11), which is intractable and requires some approximations.

The temporal edges introduce loops in the graph, which leads to an intractable inference problem. It would be possible to use an approximate method like loopy belief propagation, whose complexity is exponential in the size of the maximal clique in the graph [113]. We have found such a strategy too slow to be practical for pose estimation. Instead, we propose a two-stage approach.

The first step consists of generating a set of candidate poses in each frame. We achieve this by minimizing an approximation of (2.11) in combination with the n-best algorithm [128]. Specifically, we build on the approach of [128] by introducing frame-frame temporal smoothness among some of the body parts. In the second step, we decompose the candidate poses into limbs, and generate limb sequences across time. We then recompose the complete, accurate pose my mixing these body-part sequences. This strategy shows a better performance than simply optimizing (2.11) over the candidate poses as in [128] because it explores a larger set of poses. We now detail these two steps.

2.2.3 Generating candidate poses

In this step, we focus on generating a set of K candidate poses in each frame I_t. One approach for this task is to use the cost $C(I_t, p_t)$ in (2.10) for estimating poses in the frame I_t, and compute the K *best* and diverse solutions, as proposed in [128]. In other words, we find diverse pose configurations that yield low cost in each frame independently, regardless of the temporal smoothness. We have observed that this strategy tends to be inaccurate for parts that are difficult to estimate, such as wrists, as shown in Section 2.2.5.

We propose a method to refine the estimation of a pose p_t in a single image frame I_t using a *dummy* pose $\tilde{p}_{t+1}$ that contains only the wrist and elbow parts in the frame I_{t+1}. We define this task as optimizing the following cost function:

$$C(I_t, p_t) + \tilde{C}(I_{t+1}, \tilde{p}_{t+1}) + \tilde{\lambda}_1 \sum_{u \in \mathcal{W}} \|\tilde{p}^u_{t+1} - p^u_t - f_t(p^u_t)\|^2_2, \tag{2.13}$$

where $\mathcal{W} \subset \mathcal{V}$ represents the left and right wrists and elbows, $\tilde{\lambda}_1$ is a regularization parameter. The cost $\tilde{C}$ is defined as C in (2.10), except that only terms corresponding to wrists and elbows are considered, i.e., it contains the appearance terms ϕ_u for these parts and the deformation costs $\psi_{u,v}$ between them.

In Figure 2.7b we show the graphical model corresponding to this step. It contains two isolated loops—a setting where exact inference can be performed with loopy belief propagation [187]. This algorithm proceeds as a sequence of message passing steps. In each step, a message M^u_v is passed from node u to node v in the graph. It is then used to

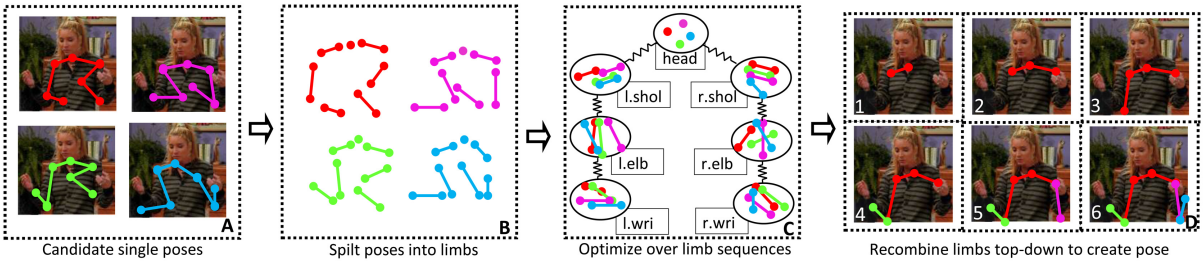


Figure 2.8 – Illustration of our limb recombination scheme. From left to right: Block-A: An image and four candidate poses, where only a part of each pose is well-aligned with the person. Block-B: We divide each candidate pose into limb parts. Block-C: We allow the recombination of limbs from different pose candidates with constraints between two limbs that have a joint in common. Block-D: An example where recombination builds an accurate pose, which is not in the original candidate set. See text for more details.

update the message from v to other nodes it is connected to. This procedure is repeated for all the nodes until convergence, i.e., none of the messages change after an update iteration is performed. On our graph, we begin by sending messages from the leaf nodes to the root, and then from the root node to the rest. After convergence, we assign each node to the label corresponding to the minimum marginal at that node. This procedure can be implemented efficiently on our graph with the distance transform technique [51].

As shown in the experiments, our approach for generating candidate poses by minimizing (2.13) instead of simply minimizing $C(I_t, p_t)$ performs better with no significant increase in computational cost.

2.2.4 Recombining limbs with variable splitting

After generating a set of K candidate poses for every frame (denoted by $\mathcal{P}_t$ for frame I_t), a simple strategy is to optimize our global objective function (2.11) over this set as:

$$\min_{p_t \in \mathcal{P}_t, \forall t} C(I_T, p_T) + \sum_{t=1}^{T-1} C(I_t, p_t) + \lambda_1 \theta(p_t, p_{t+1}, I_t, I_{t+1}). \tag{2.14}$$

This can be solved efficiently with dynamic programming in $\mathcal{O}(TK^2)$ operations, as done in [128] for example. However, we have observed that the constraint $p_t \in \mathcal{P}_t$ is an important limitation of such a strategy. On the one hand, it has the positive effect of making (2.14) tractable, but on the other hand, having only K different possible poses in every frame can be problematic. The $\mathcal{P}_t$ may contain a "good" candidate pose, but the method is unable to deal with situations where it is not the case, thus motivating us to propose an approximate scheme exploring a larger set of poses than $\mathcal{P}_t$.

Our main idea is to allow the recombination of limbs from candidate poses in $\mathcal{P}_t$ in order to create new poses, yielding a new set $\bar{\mathcal{P}}_t$ that is exponentially larger than $\mathcal{P}_t$. We will then minimize (2.14) approximately over $\bar{\mathcal{P}}_t$. As shown in Figure 2.8: (A) We break each pose p in $\mathcal{P}_t$ into limbs $l^{u,v} = (p^u, p^v)$, where (u, v) is in $\mathcal{E}$. (B) We decompose (2.14) into a sum of costs for every limb sequence. (C) We allow the recombination of limbs from

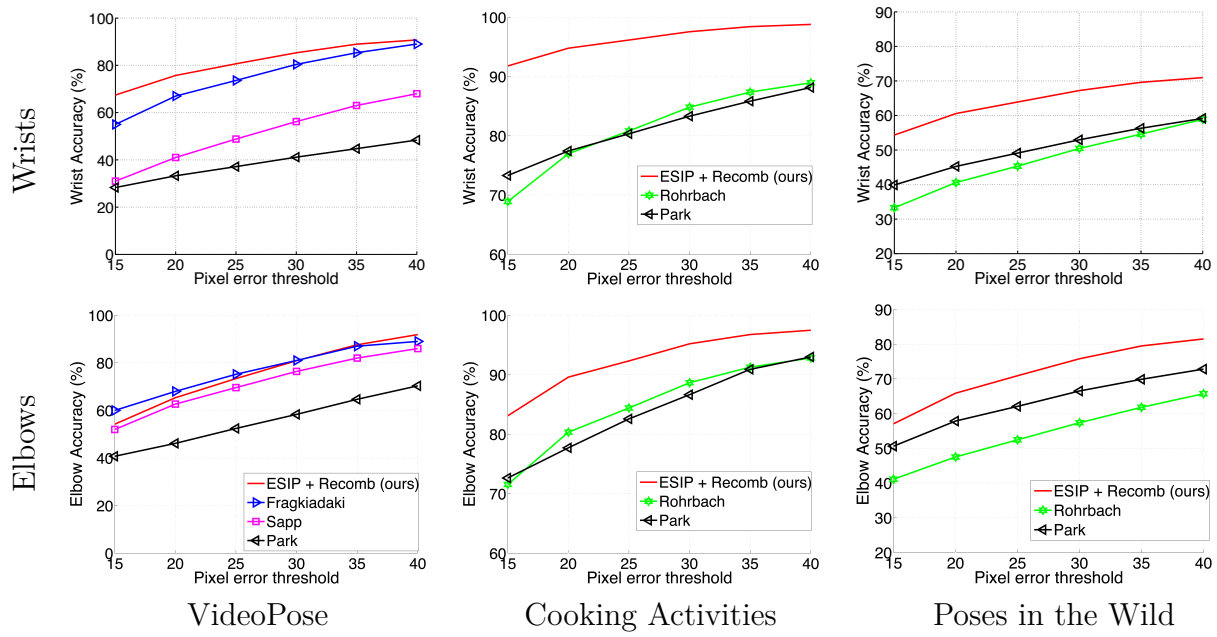


Figure 2.9 – ESIP + Recomb: We estimate poses using temporal information. Recomb refers to our method for recombining limbs to find accurate poses (§2.2.4). We compare with the following methods: Fragkiadaki [54], Park [128], Rohrbach [147], Sapp [154].

different poses as follows: consider two limbs $l^{u,v} = (p^u, p^v)$ and $l^{v,w} = (p'^v, p'^w)$ obtained respectively from two poses p and p' in $\mathcal{P}_t$. The two limbs share the same body part v, and thus p^v should be close to p'^v, such that the two individual limbs can be considered as a good approximation of the combination (p^u, p^v, p'^w). This is achieved by adding a pairwise cost $\gamma(l^{u,v}, l^{v,w}) = \lambda_2 \|p^v - p'^v\|_2^2$ to our formulation, which can be interpreted as attaching a spring between the two limbs (Figure 2.8-C). (D) We finally estimate the pose by recombining limbs in a top-to-bottom fashion, approximately minimizing the resulting objective function.

Formally, the above approach consists of approximating the objective (2.14) by a sum of costs over the limbs:

$$\sum_{(u,v)\in\mathcal{E}} S^{u,v}(l^{u,v}_{1\dots T}) + \lambda_2 \sum_{t=1}^{T} \gamma(l_t^{\mathrm{pa}(u),u}, l_t^{u,v}), \tag{2.15}$$

where $l^{u,v}_{1\dots T}$ represents a limb sequence $(l_1^{u,v}, \dots, l_T^{u,v})$, the function γ is the cost defined in the previous paragraph, and $\mathrm{pa}(u)$ represents the parent node of the body part u in the tree. Note that to simplify the notation, we associate the head to a limb (h, h) with $\mathrm{pa}(h) = h$, where h in $\mathcal{V}$ is the root of the tree. The score $S^{u,v}(l^{u,v}_{1\dots T})$ for a limb (u, v) contains all the pairwise terms in (2.14) involving p_t^u and p_t^v, as well as all the terms involving p_t^v's only. To further simplify the computations and improve the speed of the procedure, we approximate the non-temporal terms involving p_t^u and $p_t^u - p_t^v$ by the cost $C(I_t, p_t)$ computed in Section 2.2.3.

We then proceed in a top-to-bottom fashion; we start by estimating the head sequence, which is usually the most reliable body part in pose estimation, by minimizing

the corresponding function $S^{h,h}$ over the set of head candidates. This can be done in $\mathcal{O}(K^2T)$ operations with dynamic programming. In the next step, we estimate the limbs connected to the head by minimizing the cost $S^{h,v}(l^{h,v}_{1...T}) + \lambda_2 \sum_{t=1}^{T} \gamma(l_t^{\mathrm{pa}(h),h}, l_t^{h,v})$. Again, this is done in $\mathcal{O}(K^2T)$ operations. We proceed recursively until the wrists are estimated. The procedure is approximate, but turns out to be effective in practice, as shown in Section 2.2.5. It improves upon (2.14) by exploring a larger set $\bar{\mathcal{P}}_t$ instead of $\mathcal{P}_t$. Practical extensions and implementation details of our method are detailed in [A10].

2.2.5 Results: Comparison to related methods

We evaluate our method on three benchmarks: VideoPose [153], MPII Cooking Activities [147], and Poses in the Wild. Our Poses in the Wild dataset contains 30 sequences, with about 30 frames each, extracted from the Hollywood movies "Forrest Gump", "The Terminal", and "Cast Away". We manually annotated all the frames with upper-body poses. In contrast to the VideoPose and Cooking Activities datasets, it contains realistic poses in outdoor scenes, with background clutter, severe camera motion and body-part occlusions. In the following we present a selection of our results, comparing them to related approaches. A detailed evaluation is available in [A10].

As shown in Figure 2.9, our complete model (ESIP + Recomb) outperforms [54] (Fragkiadaki in the figure) by nearly 12% on the VideoPose dataset for estimating wrists (15 pixel error). On the Cooking Activities dataset, ESIP + Recomb shows 11% (elbows), 18% (wrists) and 12% (elbows), 23% (wrists) improvement over [128] (Park) and [147] (Rohrbach) respectively. In the case of the Poses in the Wild dataset, ESIP + Recomb is over 15% better than the baseline SIP model, and also shows 7% (elbows), 14% (wrists) and 16% (elbows), 21% (wrists) improvements over [128] and [147] respectively. The improvements over [128], although significant (7%, 14%), are less pronounced (compared to those on the Cooking Activities dataset), as the color-based tracking in [128] works better on our dataset.

2.2.6 Summary

Our approach for pose estimation in video consists of two steps: (i) an extended single image pose model using optical flow cues between consecutive frames, and (ii) a flexible scheme for splitting poses into limbs, generating limb sequences across time, and recomposing them to generate better poses. We also presented a new challenging dataset, Poses in the Wild, containing real-world scenarios unavailable in other datasets.

2.3 Scene Text Understanding

One of the first things we notice in the scene illustrated in Figure 2.10 is the sign board and the text it contains, which help understand the scene and provide several contextual cues. Popular recognition methods typically ignore such text, and identify other objects such as car, person, tree, and regions such as road, sky. The importance of text in images is also highlighted in the experimental study conducted by Judd et al. [77]. They found that viewers fixate on text when shown images containing text and other objects. This is further evidence that text recognition forms a useful component in understanding scenes.

In addition to being an important component of scene understanding, scene text recognition has many potential applications, such as image retrieval, auto navigation, scene text to speech systems, developing apps for visually impaired people [A17], [118]. Our method for solving this task is inspired by the many advancements made in the object detection and recognition problems [31], [35], [49], [162]. We present a framework for recognizing text that exploits bottom-up and top-down cues. The bottom-up cues are derived from individual character detections from an image. Naturally, these windows contain true as well as false positive detections of characters. We build a conditional random field (CRF) model [97] on these detections to determine not only the true positive detections, but also the word they represent jointly. We impose top-down cues obtained from a lexicon-based prior, i.e., language statistics, on the model. In addition to disambiguating between characters, this prior also helps us in recognizing words.

The first contribution of this work is a joint framework with seamless integration of multiple cues—individual character detections and their spatial arrangements, pairwise lexicon priors, and higher-order priors—into a CRF framework which can be optimized effectively. The proposed method performs significantly better than other related energy minimization based methods for scene text recognition. Our second contribution is devising a cropped word recognition framework which is applicable not only to closed vocabulary text recognition (where a small lexicon containing the ground truth word is provided with each image), but also to a more general setting of the problem, i.e., open vocabulary scene text recognition (where the ground truth word may or may not belong to a generic large lexicon or the English dictionary). The third contribution is comprehensive experimental evaluation, in contrast to many recent works, which either consider a subset of benchmark datasets or are limited to the closed vocabulary setting. We evaluate on a number of cropped word datasets (ICDAR 2003, 2011 and 2013 [3], SVT [4], and IIIT 5K-word [A18]) and show results in closed and open vocabulary settings. Additionally, we analyzed the effectiveness of individual components of the framework, the influence of parameter settings, and the use of convolutional neural network (CNN) based features [75].

Related work. The task of understanding scene text has gained a huge interest for more than a decade [42], [117], [181], [183]. It is closely related to the problem of Optical Character Recognition (OCR), which has a long history in the computer vision and pattern recognition communities [114]. However, the success of OCR systems is largely restricted to text from scanned documents. Scene text exhibits a large variability in

Figure 2.10 – A typical street scene image taken from Google Street View. It contains very prominent sign boards with text on the building and its windows. It also contains objects such as car, person, tree, and regions such as road, sky. Many scene understanding methods recognize these objects and regions in the image successfully, but overlook the text on the sign board, which contains rich, useful information. The goal of this work is to address this gap in understanding scenes.

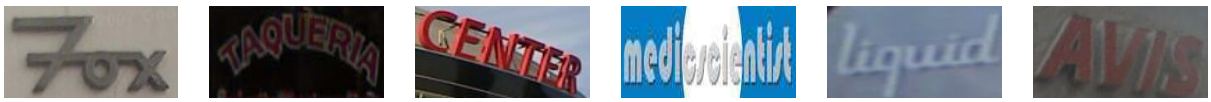


Figure 2.11 – Challenges in scene text recognition. A few sample images from the SVT and IIIT 5K-word datasets are shown to highlight the variation in view point, orientation, non-uniform background, non-standard font styles and also issues such as occlusion, noise, and inconsistent lighting. Standard OCRs perform poorly on these datasets (as seen in [117], [181]).

appearance, as shown in Fig. 2.11, and can prove to be challenging even for the state-of-the-art OCR methods (see [117], [181]). Several alternatives have started to address this challenge, through methods for localizing text characters [117], integrating character recognition and linguistic knowledge [40], [172], matching text images to a pre-defined text dictionary (lexicon) [181], or deep learning approaches [17], [74]. A more elaborate discussion on related work is available in [A16] and a survey paper [200].

2.3.1 The recognition model

We propose a conditional random field (CRF) model for recognizing words. The CRF is defined over a set of N random variables $x = \{x_i | i \in \mathcal{V}\}$, where $\mathcal{V} = \{1, 2, \ldots, N\}$. Each random variable x_i denotes a potential character in the word, and can take a label from the label set $\mathcal{L} = \{l_1, l_2, \ldots, l_k\} \cup \epsilon$, which is the set of English characters, digits and a null label ϵ to discard false character detections. The most likely word represented by the set of characters x is found by minimizing the energy function, $E : \mathcal{L}^n \rightarrow \mathbb{R}$,

Figure 2.12 – Typical challenges in character detection. (left) Inter-character confusion: A window containing parts of the two *o*'s is falsely detected as *x*. (right) Intra-character confusion: A window containing a part of the character B is recognized as E.

corresponding to the random field. The energy function E can be written as sum of potential functions:

$$E(x) = \sum_{c \in \mathcal{C}} \psi_c(x_c), \tag{2.16}$$

where $\mathcal{C} \subset \mathcal{P}(\mathcal{V})$, with $\mathcal{P}(\mathcal{V})$ denoting the powerset of $\mathcal{V}$. Each x_c defines a set of random variables included in subset c, referred to as a clique. The function ψ_c defines a constraint (potential) on the corresponding clique c. We use unary, pairwise and higher order potentials in this work, and define them in Section 2.3.1. The set of potential characters is obtained by the character detection step discussed in Section 2.3.1. The neighbourhood relations among characters, modelled as pairwise and higher order potentials, are based on the spatial arrangement of characters in the word image.

Character detection

The first step in our approach is to detect potential locations of characters in a word image. In this work we use a sliding window based approach for detecting characters. This technique has been very successful for tasks such as, face [178] and pedestrian [31] detection, and also for recognizing handwritten words using HMM based methods [15]. Although character detection in scene images is similar to such problems, it has its unique challenges. Firstly, there is the issue of dealing with many categories (63 in all) jointly. Secondly, there is a large amount of inter-character and intra-character confusion, as illustrated in Fig. 2.12.

We consider windows at multiple scales and spatial locations. The location of the ith window, d_i, is given by its center and size. The set $\mathcal{K} = \{c_1, c_2, \ldots, c_k\}$, denotes label set. Note that $k = 63$ for the set of English characters, digits and a background class (null label) in our work. Let ϕ_i denote the features extracted from a window location d_i. Given the window d_i, we compute the likelihood, $p(c_j|\phi_i)$, of it taking a label c_j for all the classes in $\mathcal{K}$. In our implementation, we used explicit feature representation [176] of histogram of gradient (HOG) features [31] for ϕ_i, and the likelihoods p are (normalized) scores from a one vs rest multi-class support vector machine (SVM). This basic sliding window detection approach produces many potential character windows, but not all of them are useful for recognizing words. We discard some of the weak detection windows with a pruning step [A16].

Graph construction and energy formulation

We solve the problem of minimizing the energy function (2.16) on a corresponding graph, where each random variable is represented as a node in the graph. We begin by ordering the character windows based on their horizontal location in the image, and add one node each for every window sequentially from left to right. The nodes are then connected by edges. Since it is not natural for a window on the extreme left to be strongly related to another window on the extreme right, we only connect windows which are close to each other. The intuition behind close-proximity windows is that they could represent detections of two separate characters. As we will see later, the edges are used to encode the language model as top-down cues. Such pairwise language priors alone may not be sufficient in some cases, for example, when an image-specific lexicon is unavailable. Thus, we also integrate higher order language priors in the form of n-grams computed from the English dictionary by adding an auxiliary node connecting a set of n character detection nodes.

Each (non-auxiliary) node in the graph takes one label from the label set $\mathcal{L} = \{l_1, l_2, \ldots, l_k\} \cup \epsilon$. Recall that each l_u is an English character or digit, and the null label ϵ is used to discard false windows that represent background or parts of characters. The cost associated with this label assignment is known as the unary cost. The cost for two neighbouring nodes taking labels l_u and l_v is known as the pairwise cost. This cost is computed from bigram scores of character pairs in the English dictionary or an image-specific lexicon. The auxiliary nodes in the graph take labels from the extended label set $\mathcal{L}_e$. Each element of $\mathcal{L}_e$ represents one of the n-grams present in the dictionary and an additional label to assign a constant (high) cost to all n-grams that are not in the dictionary. The proposed model is illustrated in Fig. 2.13, where we show a CRF of order four as an example. Once the graph is constructed, we compute its corresponding cost functions as follows.

Unary cost. The unary cost of a node taking a character label is determined by the SVM confidence scores. The unary term ψ_1, which denotes the cost of a node x_i taking label l_u, is defined as:

$$\psi_1(x_i = l_u) = 1 - p(l_u|x_i), \tag{2.17}$$

where $p(l_u|x_i)$ is the SVM score of character class l_u for node x_i, normalized with Platt's method [135]. The cost of x_i taking the null label ϵ is defined with the intuition that detected windows should have a high classifier confidence and their aspect ratio should agree with that of the corresponding character in the training data.

Pairwise cost. The pairwise cost of two neighbouring nodes x_i and x_j taking a pair of labels l_u and l_v respectively is determined by the cost of their joint occurrence in the dictionary. This cost ψ_2 is given by:

$$\psi_2(x_i = l_u, x_j = l_v) = \lambda_{\text{l}} \exp(-\beta p(l_u, l_v)), \tag{2.18}$$

where $p(l_u, l_v)$ is the score determining the likelihood of the pair l_u and l_v occurring together in the dictionary. The parameters λ_{l} and β are set empirically as $\lambda_{\text{l}} = 2$ and

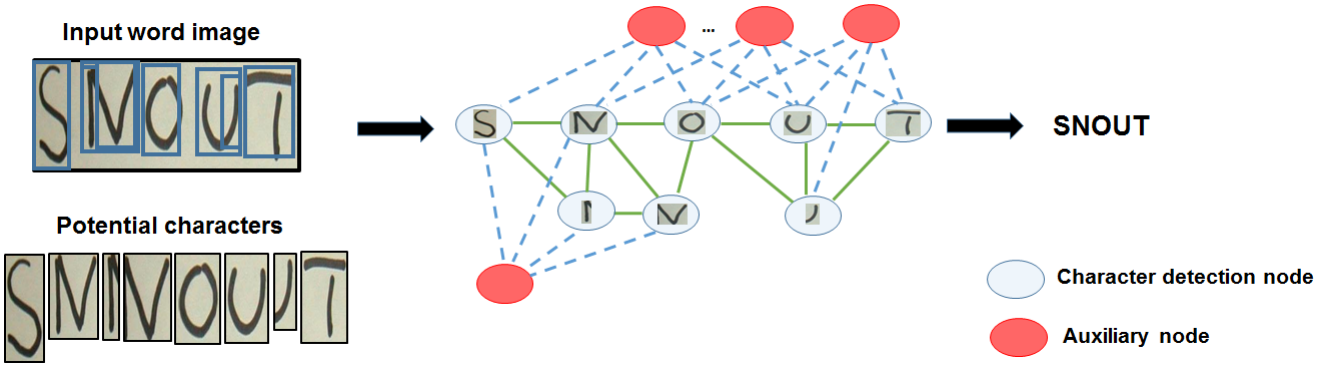


Figure 2.13 – The proposed model illustrated as a graph. Given a word image (shown on the left), we evaluate character detectors and obtain potential character windows, which are then represented in a graph. These nodes are connected with edges based on their spatial positioning. Each node can take a label from the label set containing English characters, digits, and a null label (to suppress false detections). To integrate language models, i.e., n-grams, into the graph, we add auxiliary nodes (shown in red), which constrain several character windows together (sets of 4 characters in this example). Auxiliary nodes take labels from a label set containing all valid English n-grams and an additional label to enforce high cost for an invalid n-gram.

$\beta = 50$ in all our experiments. The score $p(l_u, l_v)$ is commonly computed from joint occurrences of characters in the lexicon [40], [172]. This prior is effective when the lexicon size is small, but it is less so as the lexicon increases in size. Furthermore, it fails to capture the location-specific information of pairs of characters. As a toy example, consider a lexicon with only two words CVPR and ICPR. Here, the character pair (P,R) is more likely to occur at the end of the word, but a standard bigram prior model does not incorporate this location-specific information.

To overcome the lack of location-specific information, we devise a node-specific pairwise cost by adapting [145] to the scene text recognition problem. We evaluated our approach with both the pairwise terms, and found that the node-specific prior achieves better performance. The cost of either node taking the null label is defined accordingly [A16], to ensure that when two character windows overlap significantly, only one of them is assigned a character/digit label in order to avoid parts of characters being labelled.

Higher order cost. An auxiliary node corresponding to every clique of size k (where k is more than 2) is added to represent the k-th order cost in the graph. The higher order cost is then decomposed into unary and pairwise terms with respect to this node, similar to [151]. Each auxiliary node in the graph takes one of the labels from the extended label set $\{L_1, L_2, \ldots, L_M\} \cup L_{M+1}$, where labels $L_1 \ldots L_M$ represent all the k-grams in the dictionary. The additional label L_{M+1} denotes all those k-grams which are absent in the dictionary. The unary cost for an auxiliary variable and the pairwise costs between an auxiliary node and a non-auxiliary node are detailed in [A16].

Computing language priors. We compute n-gram based priors from the lexicon (or dictionary) and then adapt standard techniques for smoothing these scores [172] to the open and closed vocabulary cases. Image-specific lexicons (small or medium) are used in the closed vocabulary setting, while in the open vocabulary case we use a lexicon containing half a million words (henceforth referred to as large lexicon) provided by [183] to compute these scores.

Inference. Having computed the unary, pairwise and higher order terms, we use the sequential tree-reweighted message passing (TRW-S) algorithm [89] to minimize the energy function. The TRW-S algorithm maximizes a concave lower bound of the energy. It begins by considering a set of trees from the random field, and computes probability distributions over each tree. These distributions are then used to reweight the messages being passed during loopy belief propagation [131] on each tree. The algorithm terminates when the lower bound cannot be increased further, or the maximum number of iterations has been reached.

In summary, given an image containing a word, we: (i) locate the potential characters in it with a character detection scheme, (ii) define a random field over all these potential characters, (iii) compute the language priors and integrate them into the random field model, and then (iv) infer the most likely word by minimizing the energy function corresponding to the random field.

2.3.2 Datasets and evaluation protocols

Several public benchmark datasets for scene text understanding have been released in recent years. ICDAR [3] and Street View Text (SVT) [4] datasets are two of the initial datasets for this problem. They both contain data for text localization, cropped word recognition and isolated character recognition tasks. In this paper we use the cropped word recognition part from these datasets. Although these datasets have served well in building interest in the scene text understanding problem, they are limited by their size of a few hundred images. To address this issue, we introduced the IIIT 5K-word dataset [A18], containing a diverse set of 5000 words.

We evaluate the word recognition accuracy in two settings: closed and open vocabulary. Following previous work [A18], [158], [181], we evaluate case-insensitive word recognition on SVT, ICDAR 2003, IIIT 5K-word, and case-sensitive word recognition on ICDAR 2011 and ICDAR 2013. For the closed vocabulary recognition case, we perform a minimum edit distance correction, since the ground truth word belongs to the image-specific lexicon. On the other hand, in the case of open vocabulary recognition, where the ground truth word may or may not belong to the large lexicon, we do not perform edit distance based correction.

2.3.3 Results: Open vocabulary recognition

In this manuscript we highlight experimental results in the open vocabulary setting, where we use a lexicon of 0.5 million words from [183] instead of image-specific lexicons

(known as closed vocabulary setting) to compute the language priors. The remaining experimental analyses are available in [A16]. Many character pairs are equally likely in this large lexicon setting, thereby rendering pairwise priors less effective than in the case of a small lexicon. We use priors of order four to address this. Results on various datasets in this setting are shown in Table 2.2. We compare our method with recent work by Feild and Miller [47] on the ICDAR 2003 dataset, where our method with HOG features shows a comparable performance. Note that [47] additionally uses web-based corrections, unlike our method, where the results are obtained directly by performing inference on the higher order CRF model. On the ICDAR 2011 and 2013 datasets we compare our method with the top performers from the respective competitions. Our method outperforms the ICDAR 2011 robust reading competition winner (TH-OCR method) method by 17%. This performance is also better than the method of Weinman et al. [182].

On the ICDAR 2013 dataset, the proposed higher order model is significantly better than the baseline and is in the top-5 performers among the competition entries. The winner of this competition (PhotoOCR) uses a large proprietary training dataset, which is unavailable publicly, making it infeasible to do a fair comparison. Other methods, e.g., NESP [94], use many preprocessing techniques, followed by off-the-self OCR. Such preprocessing techniques are highly dataset dependent and may not generalize easily to all the challenging datasets we use. Despite the lack of these preprocessing steps, our method shows a comparable performance. On the IIIT 5K-word dataset, which is large (three times the size of ICDAR 2013 dataset) and challenging, the only published result to our knowledge is Strokelets [199] from CVPR 2014. Our method performs 7% better than Strokelets. Using CNN features instead of HOG further improves our word recognition accuracy, as shown in Table 2.2.

2.3.4 Summary

Our model combines bottom-up cues from character detections and top-down cues from lexicon. We jointly infer the location of true characters and the word they represent as a whole. We evaluated our method on several challenging street scene text datasets, and showed that our approach significantly advances the energy minimization based paradigm for scene text recognition. We also showed that this is complementary to the resurgence of convolutional neural network based techniques, which can help build better scene understanding systems.

Table 2.2 – Word recognition accuracy (in %): open vocabulary setting. The results of our higher order model ("This work") with HOG and CNN features are presented here. Since the network used here to compute CNN features, i.e. [75], is learnt on data from several sources (e.g., ICDAR 2013), we evaluated with CNN features only on ICDAR 2003 and IIIT-5K word datasets, as recommended by the authors [75]. We also compare with top performers (based on [80], [156]) in the ICDAR 2011 and 2013 robust reading competitions. We follow standard protocols for evaluation (see Section 2.3.2).

Method	Accuracy
ICDAR 2003 dataset	
Baseline (ABBYY)	46.51
Baseline (CSER+tesseract) [58]	50.99
Feild and Miller [47]	62.76
Our variants	
Pairwise [A19]	50.99
Higher order [This work, HOG]	63.02
Higher order [This work, CNN]	67.67
ICDAR 2011 dataset	
Baseline (ABBYY)	46.00
Baseline (CSER+tesseract) [58]	51.98
Weinman et al. [182]	57.70
Feild and Miller [47]	48.86
ICDAR'11 competition [156]	
TH-OCR System	41.20
KAIST AIPR System	35.60
Neumann's Method	33.11
Our variants	
Pairwise [A19]	48.11
Higher order [This work, HOG]	58.03
ICDAR 2013 dataset	
Baseline (ABBYY)	45.30
Baseline (CSER+tesseract) [58]	50.26
ICDAR'13 competition [80]	
PhotoOCR [17]	82.83
NESP [94]	64.20
PicRead [122]	57.99
POINEER [182], [183]	53.70
Field's Method [47]	47.95
TextSpotter [117], [119], [120]	26.85
Our variants	
Pairwise [A19]	49.86
Higher order [This work, HOG]	60.18
IIIT 5K-Word	
Baseline (ABBYY)	14.60
Baseline (CSER+tesseract) [58]	25.00
Stroklets [199]	38.30
Our variants	
Pairwise [A19]	32.00
Higher order [This work, HOG]	44.50
Higher order [This work, CNN]	46.73

Chapter 3

Motion Understanding

This chapter is based on the following publications.

— Y. Hua, K. Alahari, and C. Schmid,
"Online Object Tracking with Proposal Selection," ICCV, 2015 [PDF]

— J. Lezama, K. Alahari, J. Sivic, and I. Laptev,
"Track to the Future: Spatio-temporal Video Segmentation with Long-range Motion Cues," CVPR, 2011 [PDF]

— P. Tokmakov, C. Schmid, and K. Alahari,
"Learning to Segment Moving Objects," *International Journal on Computer Vision*, 2018 (In press) [PDF]

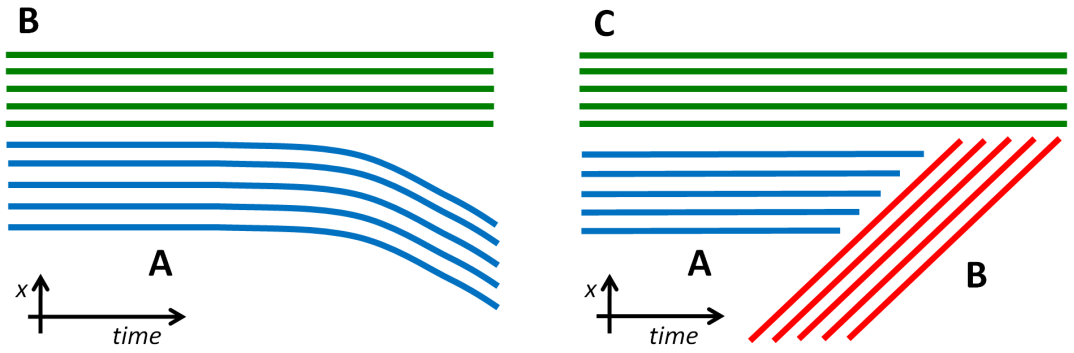
This chapter presents a selection of approaches for understanding motion. We begin with the work on extracting long-range motion cues, in the form of occlusion-aware trajectories, from videos (Section 3.1). This is a form of low-level understanding of motion for video segmentation. The second approach addresses the problem of localizing objects undergoing transformations, such as rotation, in a video sequence. Here we consider the semi-supervised case, where only the first frame is annotated with the object of interest (Section 3.2). These two works are examples where motion cues are hand-crafted into the framework. The third approach presented in this chapter shows how motion features can be learned directly from synthetic data (Section 3.3).

3.1 Long-Range Motion Cues for Segmentation

We seek to develop an *intermediate representation*, which exploits long-range temporal cues available in the video, and thus provides a stepping stone towards automatic interpretation of dynamic scenes. In particular, we aim to obtain a spatio-temporal over-segmentation of video that respects object boundaries, and at the same time temporally associates (subsets of) object pixels whenever they appear in the video. This is a challenging task, as local image measurements often provide only a weak cue for the presence of object boundaries. At the same time, object appearance may significantly change over the frames of the video due to, for example, changes in the camera viewpoint, scene illumination or object orientation. While obtaining a complete segmentation of all objects in the scene may not be possible without additional supervision, we propose to partially address these challenges in this work.

We combine local image and motion measurements with *long-range motion cues* in the form of carefully grouped point-tracks, which extend over many frames in the video. Incorporating these long point-tracks into spatio-temporal video segmentation brings three principal benefits: (i) pixel regions can be associated by point-tracks over many frames in the video; (ii) locally similar motions can be disambiguated over a larger frame baseline; and (iii) motion and occlusion events can be propagated to frames with no object/camera motion.

Related Work. Individual frames in a video can be segmented independently using existing single image segmentation methods [30], [48], [160], but the resulting segmentation is not consistent over consecutive frames. Video sequences can also be segmented into regions of locally coherent motion by analyzing dense motion fields [159], [186] in neighbouring frames. Methods such as [34], [62], [168], [175], [204] rely on cues computed at a local level for segmentation. We build on the hierarchical, graph-based segmentation method of Grundmann et al. [62] and extend it by incorporating long-range motion cues into the segmentation. Other related work based on layered representation of video, e.g., [95], is limited to using a locally affine parametric model. Our method groups point trajectories, following the approach in [23], but incorporates additional occlusion constraints, and recovers partial depth ordering between groups of tracks. A more detailed presentation of related work is available in [A15].



(a) (b)

Figure 3.1 – A toy example illustrating x-t video slices. (a): Objects A and B can be separated early in time if we propagate their motion from a latter point in time. (b): Objects A and C can be separated despite their motion similarity if we take the relative depth ordering, as induced by occlusions, into account. Here, A is occluded by B, and B is occluded by C.

3.1.1 Track clustering with occlusion reasoning

Long-term motion can provide strong low-level cues for many vision tasks. For example, two static objects can be separated based on their past or future independent motion if this motion evidence is propagated over time (see Figure 3.1(a)). Similarly, if an object B occludes A while C occludes B (as shown in Figure 3.1(b)), we can reason that A and C are in fact two different objects based on the relative depth ordering. Such analyses can also be very useful for higher-level scene understanding in terms of objects and event categories. For example, a point cloud which appears at the door and descends to a chair is very likely to belong to a sitting person who just entered a room.

Our goal in this work is to over-segment the video into groups of pixels belonging to the same object over time and to provide novel building blocks for higher level video interpretation. We start reasoning about the long-term motion by clustering a sparse set of point-tracks extracted from many frames of a video [23]. Our method, however, is not specific to this particular choice and could be used with other point tracking algorithms such as KLT [174] or Particle Video [152].

Depth order based track clustering. We propose a novel energy-based method for track clustering, which both: (i) groups tracks together based on their similarity, and (ii) establishes a relative depth ordering between track clusters based on the occlusion and disocclusion relations among tracks. As a direct consequence, the method is able to separate tracks, which have similar motion, based on their depth ordering, provided there is sufficient occlusion evidence in the video.

We represent each track with a random variable X_i, which takes a label $x_i \in \mathcal{L}$, representing its cluster assignment. The label set $\mathcal{L} = \{1, 2, \ldots, c\}$, denotes the set of

clusters. We select the number of labels manually. However, an automatic selection of number of labels can be addressed with a label cost term, such as in [33]. Let n be the number of tracks in the video sequence, and $\mathbf{X} = \{X_1, X_2, \ldots, X_n\}$ be the set of random variables. A labelling $\mathbf{x}$ refers to any possible assignment of labels to the random variables, and takes values from the set $\mathbf{L} = \mathcal{L}^n$. We define the cost of a label assignment $E(\mathbf{x})$, i.e., $E(\mathbf{X} = \mathbf{x})$, as follows:

$$E(\mathbf{x}) = \sum_{(i,j)\in\mathcal{E}} \Big[\alpha_{ij}\phi_1(x_i, x_j) + (1 - \alpha_{ij})\phi_2(x_i, x_j) + \gamma_{ij}\phi_3(x_i, x_j)\Big], \tag{3.1}$$

where $\mathcal{E}$ is the set of pairs of interacting tracks. As detailed later, α_{ij} and γ_{ij} measure, respectively, the motion similarity and occlusion cost for a pair of tracks (i, j). The three potentials, $\phi_1(x_i, x_j)$, $\phi_2(x_i, x_j)$, and $\phi_3(x_i, x_j)$ model the joint cost of labelling tracks i and j. Interacting tracks, $\mathcal{E}$, are only required to have a temporal overlap of at least two frames, i.e., any two tracks in the video sequence can potentially interact. This would be equivalent to a generalization of the four or the eight neighbourhood used in standard pixel-wise Markov random field models [20]. Note that since our goal is to generate an unsupervised segmentation of a given video sequence, this energy function $E(\mathbf{x})$ does not include any unary terms, which model the cost of assigning labels to each random variable independently. The cost function can be extended to incorporate unary potentials, similar to those in [A14], [162].

Similarity constraints. To assign similar tracks to the same clusters, we use potential $\phi_1(x_i, x_j)$ which takes the form of a standard Potts model [18], [20], i.e.,

$$\phi_1(x_i, x_j) = \begin{cases} 0 & \text{if } x_i = x_j, \\ 1 & \text{otherwise.} \end{cases} \tag{3.2}$$

The intuition for including ϕ_1 is that two tracks with similar motion and close locations are more likely to belong to the same object, and thus should be assigned the same label. This "attraction" force is modulated by α_{ij} which measures the similarity between two tracks, and ensures that only similar tracks are constrained to take the same label. The values α_{ij} are computed using the distance between the time-corresponding spatial coordinates of track points $\mathbf{a}_i, \mathbf{a}_j$, as well as the distance between time-corresponding velocity values in both tracks $\mathbf{v}_i, \mathbf{v}_j$ as follows: [1]

$$\alpha_{ij} = exp\left(-\frac{(1 + ||\mathbf{a}_i - \mathbf{a}_j||_2)^2 ||\mathbf{v}_i - \mathbf{v}_j||_2^2}{2 l_{ij} \sigma_s^2}\right), \tag{3.3}$$

where l_{ij} is the length of the temporal overlap (in frames) between the tracks i and j, and σ_s is a parameter. Note that this track similarity measure is similar to the one used

1. Note that $\mathbf{a}$ is composed of (x, y) coordinates of all points in a track. When comparing two tracks i and j, we use only points from frames where both tracks overlap in time. Similarly, $\mathbf{v}_i, \mathbf{v}_j$ are composed of velocity values $x_t - x_{t-1}$ and $y_t - y_{t-1}$ for frames t and $t - 1$ for time-overlapping segments of tracks i and j.

in [23]. Furthermore, the effect of using α_{ij} with ϕ_1 is equivalent to the contrast-sensitive Potts model [20].

We also use the potential ϕ_2, which acts as a "repelling" force for tracks that have low similarity values, i.e., $(1 - \alpha_{ij})$ is high. This potential is defined as:

$$\phi_2(x_i, x_j) = \begin{cases} 1 & \text{if } x_i = x_j, \\ 0 & \text{otherwise.} \end{cases} \tag{3.4}$$

The potential ϕ_2 ensures that dis-similar tracks take different labels. The repelling force of ϕ_2 prevents the trivial solution of all tracks being assigned to the same cluster.

The effect of using ϕ_1 and ϕ_2 potentials together with the similarity measure α_{ij} is illustrated in Figure 3.2(c). As can be seen, minimization of the energy defined in terms of ϕ_1, ϕ_2 only separates the tracks of a moving person from the rest of the tracks originating from the static scene.

Depth ordering constraints. Although results in Figure 3.2(c) are consistent with our expectations, the obtained grouping of tracks is far from being perfect since the method fails to separate different objects sharing similar motion, for example, the static person in the foreground and its background. We address this problem by reasoning about occlusions and disocclusions. We observe that tracks of object A are usually terminated abruptly by the tracks of another object B if B occludes A (we denote this by $B \rightarrow A$). Moreover, $C \rightarrow B \rightarrow A$ provides evidence that C and A most likely belong to different depth layers, i.e., $C \rightarrow A$, and therefore C and A should be separated even if they share similar motion (or appearance, or both). Indeed, the case in Figure 3.1(b) corresponds to the real example in Figure 3.2, where the background (A) is being occluded by the moving person (B), which in turn is occluded by the sitting person in the front (C). In the following we integrate the notion of occlusion within our clustering framework to enable both improved clustering of tracks and inference of the relative depth ordering between track clusters.

We introduce the potential $\phi_3(x_i, x_j)$ imposing the order on the track labels as follows:

$$\phi_3(x_i, x_j) = \begin{cases} 1 & \text{if } x_i \geq x_j, \\ 0 & \text{if } x_i < x_j. \end{cases} \tag{3.5}$$

This potential is modulated by γ_{ij} in (3.1) which takes a high value for a pair of tracks (i, j), if j occludes i, i.e., $j \rightarrow i$. To measure γ_{ij} from the data, we consider a pair-wise asymmetric cost between end-points of track i and all tracks j, close to i. As can be seen from the x-t slice of the video in Figure 3.2, occlusions often result in "T-junctions" and tracks are terminated by other near-by tracks with different motion. Hence, we define score of γ_{ij} in terms of the difference in velocity $\mathbf{v}_i - \mathbf{v}_j$ and the difference in position D (see Figure 3.3) between tracks (i, j) at the moment of termination (or start) of a track i as follows:

$$\gamma_{ij} = 1 - exp\left(-\frac{d||\mathbf{v}_i - \mathbf{v}_j||_2^2}{\sigma_o^2}\right), \tag{3.6}$$

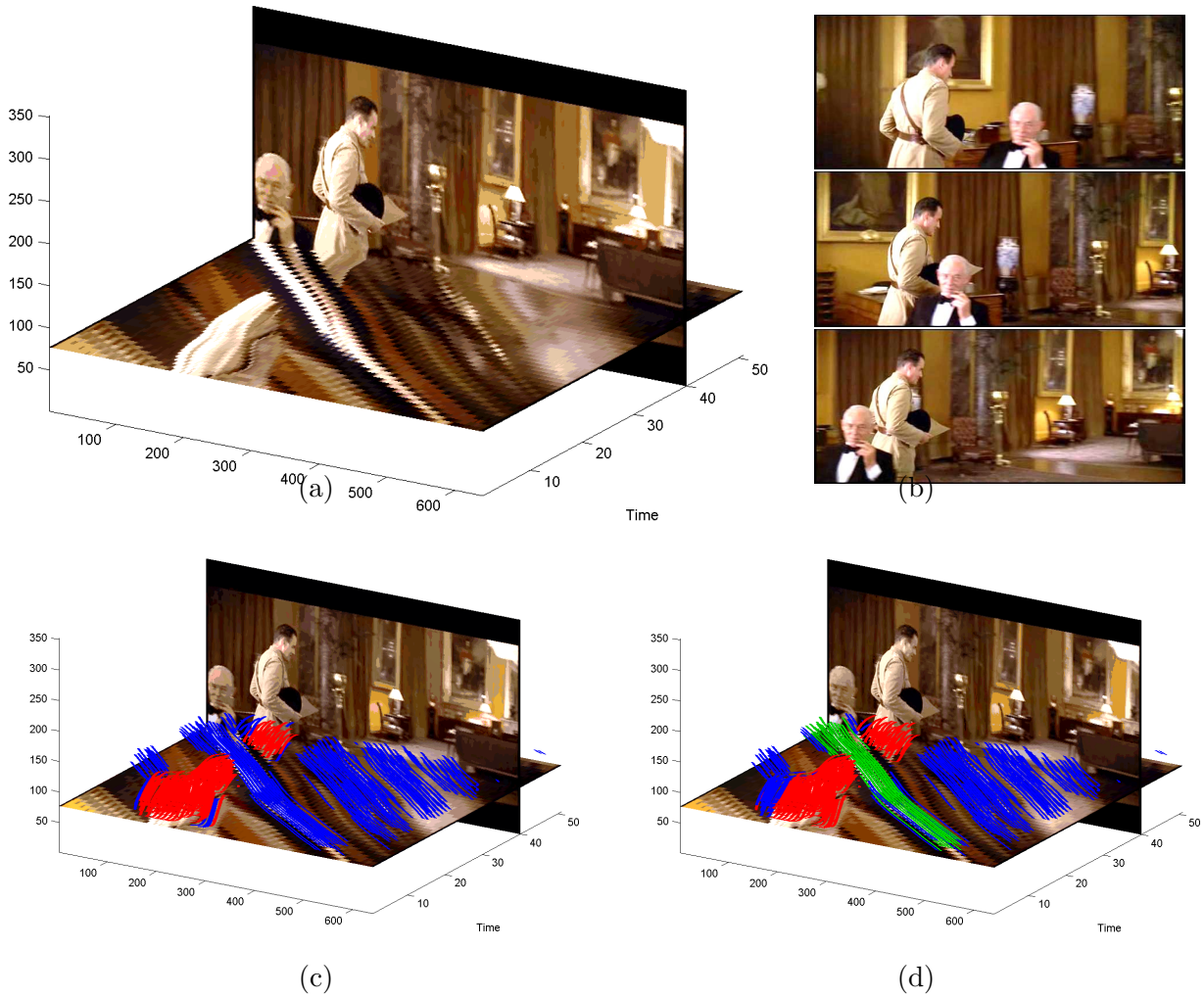


Figure 3.2 – **Track clustering.** (a)-(b): Illustration of original image sequence with a person passing behind another sitting person. On the space-time slice (a), note the two sets of "T-junctions" generated by the motion boundaries between: (i) the walking person and the background; and (ii) the walking person and the static person in the front. (c) **Results of similarity-based track clustering** using energy function with potentials ϕ_1 and ϕ_2 only (see text). (d) **Results of similarity and occlusion-based track clustering** using the full energy in (3.1). The introduction of the occlusion potential ϕ_3 to (3.1) enables separation of the static person (green) from the static background (blue). The correct relative depth ordering of track labels (from back to front: $1 < 2 < 3$) is resolved by occlusion reasoning: 1 – background (blue); 2 – moving person (red); 3 – static person in the front (green). Note that a subset of the original tracks is shown to make the figure readable. A single cluster is obtained for the walking person (red) as there are other tracks which connect the two seemingly different components, i.e., a partial occlusion. (*Best viewed in colour.*)

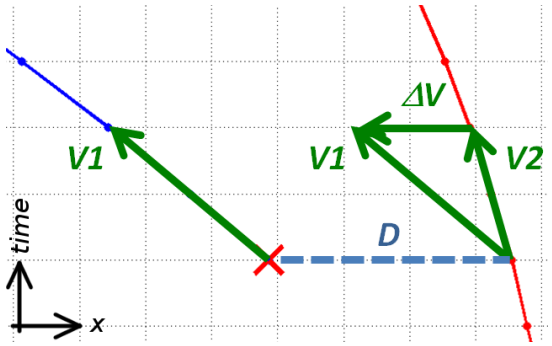


Figure 3.3 – Illustration of measurements involved in the computation of occlusion/disocclusion score γ_{ij} in (3.6). The velocity vector V_1 at the end-point of the occluded track (blue) is compared to velocity vector V_2 of a near-by track (red). D is the image distance between two tracks.

where σ_o is a parameter and d is a damping coefficient decreasing the occlusion score with the increasing distance D between tracks (i, j): $d = exp(-D^2/\sigma_d^2)$ for some parameter value σ_d. Figure 3.4 illustrates the computed values of $\gamma_{i\cdot}$ for each end-point of track i maximized over all other tracks j.

We next plug-in the potential $\phi_3(x_i, x_j)$ and the occlusion score γ_{ij} into (3.1) and optimize the full energy. The resulting clustering of the tracks is illustrated in Figure 3.2(d). In contrast to result in Figure 3.2(c), we observe the tracks of the sitting person (green) have now been separated from the background (blue). Moreover, the relative depth ordering of the labels has been correctly recovered as green → red → blue. More results for the track clustering as well as for its application to the dense video segmentation are shown in Section 3.1.3.

Optimizing the energy function. The most probable or Maximum a Posteriori (MAP) labelling $\mathbf{x}^*$ of the energy function in (3.1) is defined as: $\mathbf{x}^* = \arg\min_{\mathbf{x}\in\mathbf{L}} E(\mathbf{x})$. Note that our energy function does not satisfy the submodularity condition [88] even for the two-label case. Many methods have been proposed to solve such non-submodular energy functions [19], [89], [90]. We use the sequential tree-reweighted message passing (TRW-S) algorithm [89] because of its efficiency and accuracy for our clustering problem. The labels thus obtained for the tracks are then used to define long-range motion cues for video segmentation.

3.1.2 Graph-based video segmentation with long-range motion cues

In this section we describe our spatio-temporal video segmentation algorithm, which incorporates long-range motion cues in the form of clusters of point tracks obtained as described in Section 3.1.1. Inspired by the graph-based single image segmentation approach of Felzenszwalb and Huttenlocher [48] and its hierarchical extension to video [62],

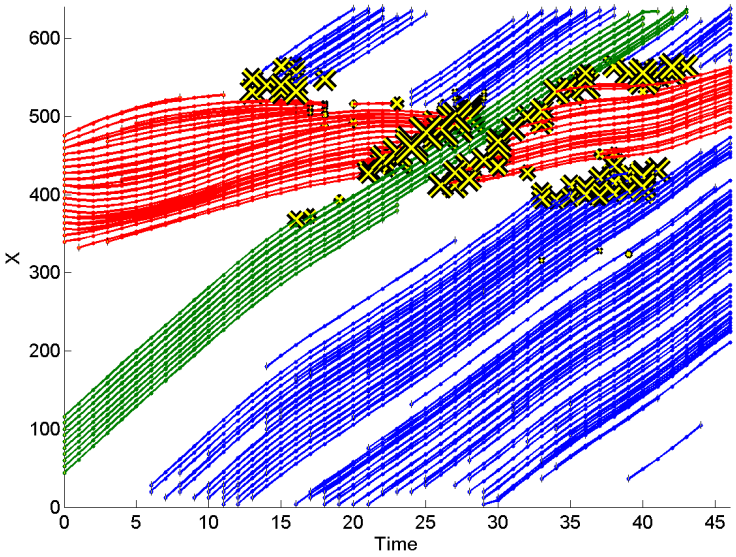
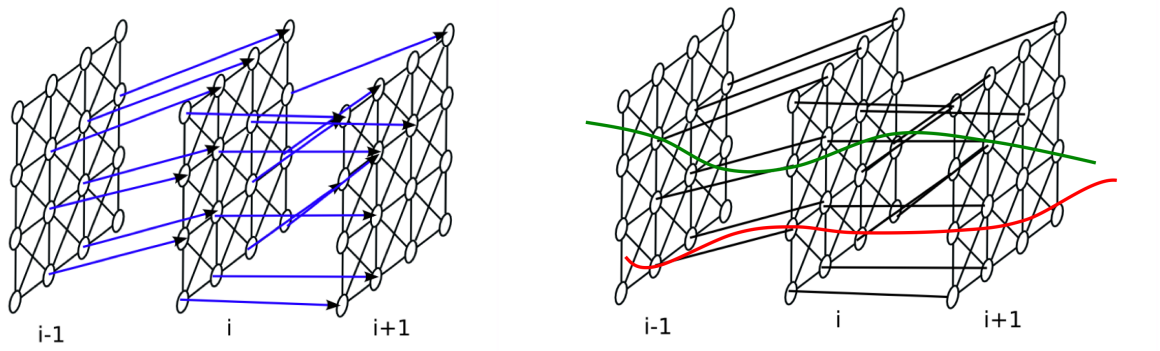


Figure 3.4 – The values of occlusion scores $\gamma_{i\cdot}$ are illustrated for end-points of all tracks extracted on the example video in Figure 3.2. Large size crosses correspond to large values of $\gamma_{i\cdot}$ and illustrate good prediction of track occlusion in this example.

we present an efficient way to incorporate long-range motion cues as additional merging constraints.

Review of the graph-based segmentation [48]. An image is represented by a graph $G = (U, E)$ with vertices $u_i \in U$ and edges $(u_i, u_j) \in E$. The elements of U are pixels and the elements of E are the edges linking neighbouring pixels. Each edge is assigned a weight, which is a measure of the colour dissimilarity between the two pixels linked by that edge. Edges are then ordered by their weight in a non-decreasing order. The ordered list of edges is traversed one by one, deciding if two pixel regions C_1 and C_2 connected by the edge considered are merged according to a score measuring the difference between C_1 and C_2, relative to the internal similarity within C_1 and C_2. Both the region difference and internal similarity are computed from the existing (pre-computed) edges in the graph and no additional measurements in the image are necessary. The algorithm is efficient as its complexity is $O(nlogn)$, where n is the number of edges in the graph.

Extensions to video [62]. The image graph in [48] can be extended to a three-dimensional graph involving all pixels in a video. We follow [62] and add an edge to each pair of neighbouring pixels in space and time. For a pixel $(x, y, t)^\top$ we add edges to its 8-connected spatial neighbours as well as to a temporal neighbour $(x + v_x, y + v_y, t + 1)^\top$, where $(v_x, v_y)^\top$ is the velocity vector estimated by optical flow at $(x, y, t)^\top$ (see Figure 3.5(a)). The weight of an edge between two pixels is set to the Euclidean distance between their colour and velocity descriptor vectors $(r, g, b, v_x, v_y)^\top$.

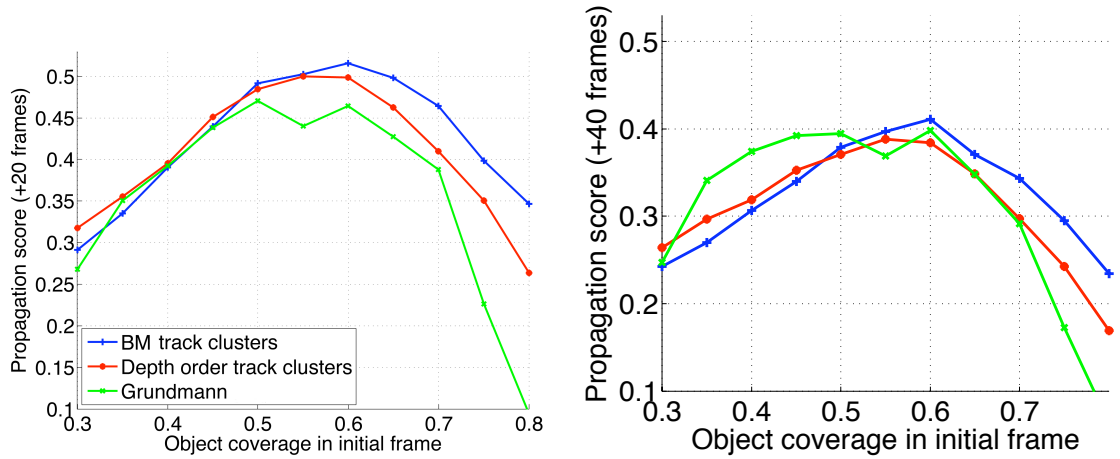


(a) (b)

Figure 3.5 – (a) **Extending the graph-based image segmenter of Felzenszwalb and Huttenlocher [48] to video**. Edges following densely estimated motion field connect pixel graphs of neighbouring frames, illustrated here for frames $i-1$, i and $i+1$. (b) **Adding long-range motion constraints:** (i) temporal edges connected by point tracks (shown as curves here) receive low cost and hence are merged first; (ii) edges connecting pixels, containing tracks belonging to different groups (shown by two different colours) receive high cost and are not considered for merging.

Grundmann et al. [62] extend the above pixel graph to a region graph within an iterative hierarchical segmentation framework. At each iteration j the graph is rebuilt by connecting neighbouring regions from the previous iteration $j-1$. Edges of the graph have weights corresponding to χ^2-distance between region histogram descriptors of colour and motion. We have compared pixel graphs with region graphs and found the latter ones to work better. We therefore build on the the hierarchical segmentation of [62] and extend it as described below.

Incorporating long-range motion cues. Ideally we wish to have a video segmentation where each object is supported by a single spatio-temporal region – an elongated volumetric segment – whose spatial support in each frame, where the object appears, should be the same as the object spatial support. However, segmenting all objects in the video from only low-level cues might be unrealistic without additional top-level supervision [150]. Hence, we aim at producing an over-segmentation, where object's "footprint" in the video is formed from a small set of spatio-temporal regions, which do not cross object boundaries. Preventing the merging of pixels that belong to different objects can be difficult or even impossible to resolve based on only local image and temporal information. In contrast, we introduce long-range information in the form of point-tracks, whose shapes provide, over longer time, additional information for disambiguating pixels with similar local appearance and motion. This is implemented in the following way. First we encourage the created spatio-temporal segments to have a long support in time by building them around point-tracks. Second, we impose constraints on the resulting segmentation based on global track clusters obtained in Section 3.1.1.



(a) +20 frames (b) +40 frames

Figure 3.6 – Propagation score vs. the object coverage in the initial frame for the proposed segmentation with long range motion cues (using depth order track clusters or BM [23] track clusters) compared to the segmentation of [62].

Encouraging long-range temporal connections. We initialize the hierarchical segmentation by building a three-dimensional pixel graph and add edges that follow point tracks. By giving these edges weight 0, we enforce the merge of all pixels along the track to one region. The tracks then act as "seeds" for the segmentation, which are then "grown" by adding additional pixels based on local similarity in motion and colour.

Segmentation with motion constraints. Here we incorporate the track clustering obtained, as described in Section 3.1.1, into the segmentation framework. Recall, that tracks are grouped according to their global motion similarity and occlusion constraints into a set of c groups. The situation is illustrated in Figure 3.5(b). The constraint is incorporated into the segmentation by introducing an additional label for each pixel (node) in the pixel graph. First, pixels along the tracks are labeled according to the cluster to which the track belongs to, a number between 1 and c. The rest of the pixels are labeled -1. Each time the algorithm is going to do a merge, it checks that either the label of the two regions is the same or one of them is -1 and it will only merge if any of these is the case. In case there is a merge, the new region will be labelled with the maximum label of the two regions being merged. This way we make sure that the regions containing tracks from different clusters will never be merged. Note that introducing motion constraints in this form does not affect the complexity of the algorithm.

While the depth ordering constraints are used to produce the track grouping, they affect the segmentation only indirectly in the form of constraints on the track clusters. However, we believe the depth ordering information will be useful for further video understanding tasks, providing useful constraints on object labels, i.e., the "wall" should appear behind "a person".

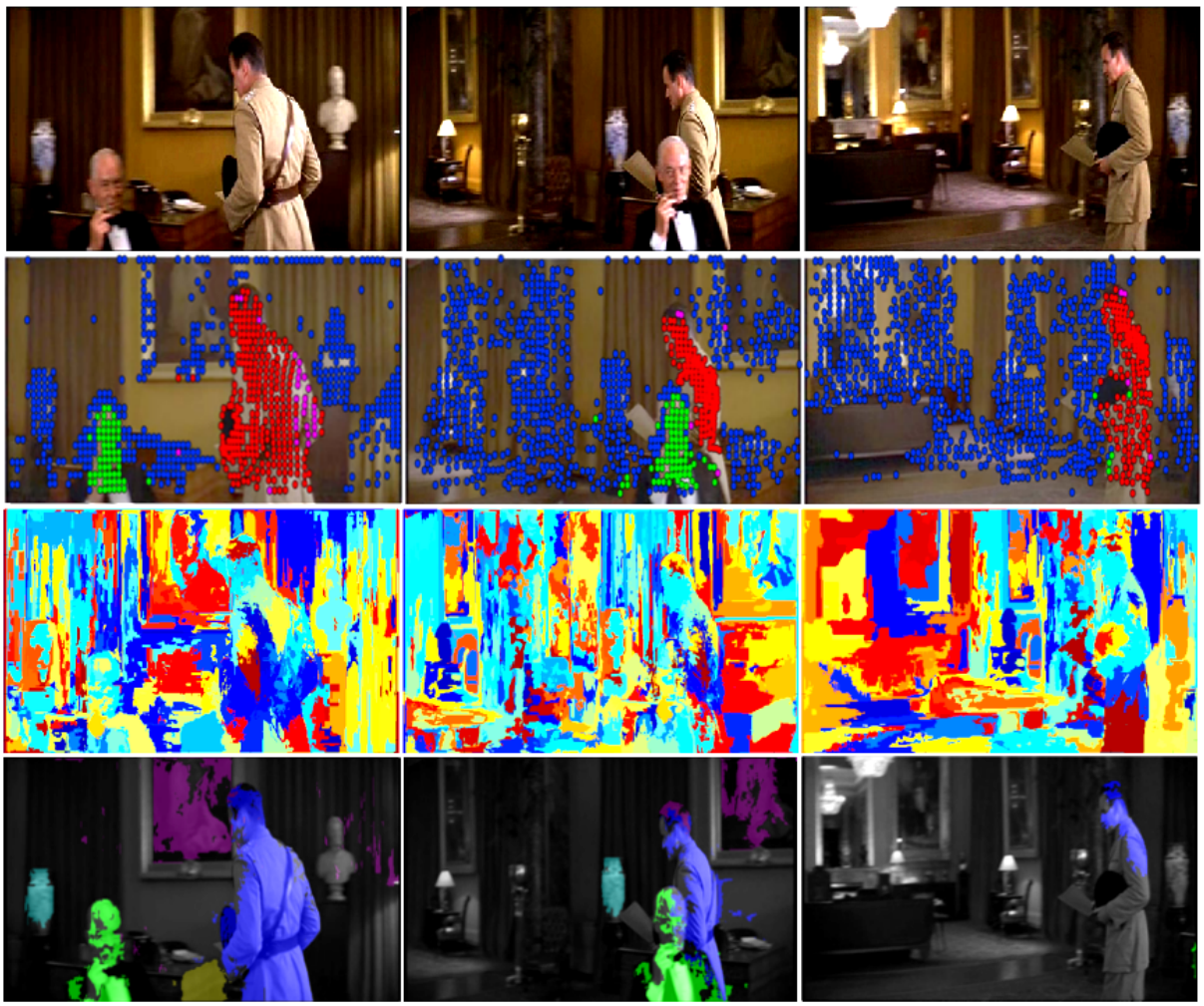


Figure 3.7 – First row: Frames 0, 20 and 40 of the original sequence, respectively. Second row: Tracked points overlaid over the video colour-coded according to the obtained clusters. The automatically inferred label ordering (front to back): green, red, magenta, and blue. Note that tracks on the sitting person (green) were correctly separated from background (blue) with the correct depth ordering, despite the fact that the person is not moving. Third row: obtained spatio-temporal over-segmentation of the video frames. Fourth row: the ground truth object regions (left) automatically propagated to the other frames (middle, right).

3.1.3 Results: Video segmentation

We use a subset of short video clips from the Hollywood 2 dataset [106] to evaluate this method. Further details are provided in [A15]. Since our goal is to have spatio-temporal regions that support objects across all frames in videos, we measure how well the obtained regions 'follow' the actual object support. For each video, we have annotated three frames (with a gap of 20 frames) with outlines of all considered objects. Then we

take one of the three annotated frames, and for each annotated object in this frame, we take the spatio-temporal regions which have a significant spatial overlap (measured by the standard intersection over union score) with the objects in that particular frame. Then we measure how well these selected regions propagate in time, by measuring the overlap of their union with the ground truth object annotations in the two other annotated frames.

Figure 3.6 shows the average propagation scores (defined formally in [A15]) obtained when propagating object regions in the 10 video clips over 20 and 40 frames. Each method produces a hierarchy of segmentations with an increasing size of segments. Hence, we plot the propagation score against the object coverage in the first frame measured by the area overlap with the ground truth. At low hierarchy levels, the (union of) small segments cover the object in the first frame well (object coverage around 0.8), but have low propagation score in time. For medium size segments (medium hierarchy levels) the quality of the segmentation in the first frame decreases (object coverage around 0.6) but the propagation in time is the best. Finally, large segments at high hierarchy levels have very low (< 0.5) object coverage and also their propagation score decreases as they often leak to background and other objects. Results are shown for the proposed video segmenter (Section 3.1.2) with tracks clustered either with the method of Section 3.1.1 (Depth order track clusters) or the track clustering method of [23] (BM [23] track clusters). Performance is compared with the popular video segmentation method of Grundmann et al. [62].

The best propagation performance is achieved by our region segmenter in combination with track clustering of [23], outperforming the method of [62] across virtually all reasonable (>0.5) object coverage levels. This demonstrates the advantage of using clustered tracks for spatio-temporal video segmentation. Using the same tracks in combination with our segmentation method in Section 3.1.1 gives somewhat worse results, but in contrast to [23], we do not do any post-processing of track clusters, and address a harder clustering task resolving depth ordering in the scene. Qualitative results of video segmentation, object region propagation and track clustering with depth ordering are illustrated in Figure 3.7. A more complete evaluation of the method is presented in [A15].

3.1.4 Summary

We have presented an efficient spatio-temporal segmentation algorithm incorporating long-range motion cues in the form of groups of point-tracks. Towards this goal, we have devised a new track clustering cost function that includes occlusion reasoning in the form of depth ordering constraints.

3.2 Online Object Tracking

Tracking-by-detection has emerged as one of the successful paradigms for tracking objects [105], [192]. It formulates the tracking problem as the task of detecting an object, which is likely to undergo changes in appearance, size, or become occluded, over time [11], [12], [65], [78], [171]. These approaches begin by training an object detector with an initialization (in the form of a bounding box) in the first frame, and then update this model over time. Naturally, the choice of exemplars used to update and improve the object model is critical [A12], [171].

Consider the *motocross* example shown in Figure 3.8. Here, the target (biker and his motorbike) undergoes several deformations as the biker performs an acrobatics routine, which leads to significant changes in the aspect ratio as well as the rotation angle of the bounding box. State-of-the-art tracking methods, e.g., [A12], [171], rely on axis-aligned bounding boxes and thus are ill-equipped to capture the accurate extents of the object in such scenarios. In this work, we aim to address this issue by treating the tracking problem as the task of selecting the best proposal containing the object of interest from several candidates. To this end, we propose a novel method to generate candidates which capture the extents of the object accurately, by estimating the transformation(s) undergone by the object.

The focus of this work is the problem of tracking a single target in monocular video sequences. The target is provided as a ground truth annotation in the first frame, as

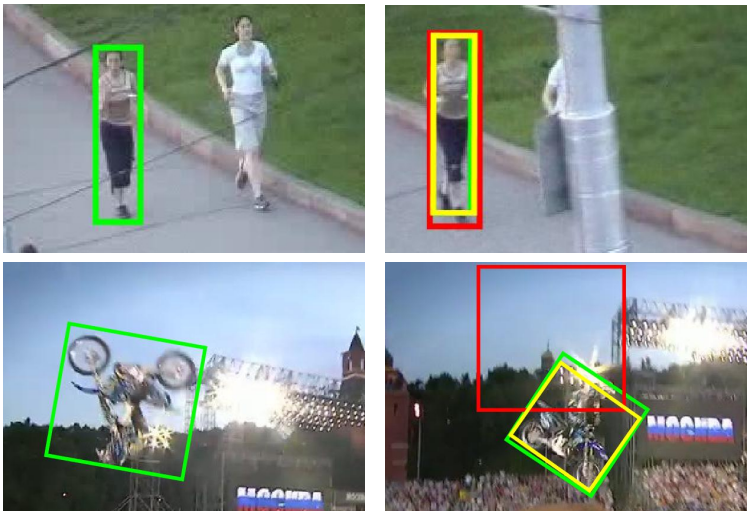


Figure 3.8 – Sample frames (cropped) from the *jogging* (top row) and *motocross* (bottom row) sequences [105]. The ground truth annotation (green) in the first frame (left) is used to train our tracker and the winner [32] of VOT2014 challenge [105]. We show these two tracking results (right) on another frame in the sequence. Our method (yellow) successfully tracks objects undergoing deformations unlike [32] (red). (*Best viewed in colour.*)

in a standard tracking-by-detection setup [65], [78]. We learn an object detector model from this annotation and evaluate it in subsequent frames to propose candidate locations that are likely to contain the target. While these proposals are sufficient to track objects undergoing a small subset of transformations over time, such as translation shown in Figure 3.8-top, they cannot handle generic transformations, e.g., similarity transformation shown in Figure 3.8-bottom. We address this problem by: (i) introducing novel additional proposals to improve the candidate location set, and (ii) using multiple cues to select the proposal that is most likely to contain the target. Additional proposals are computed by robustly estimating the parameters of similarity transformation (i.e., scaling, translation, rotation) that the target is likely to have undergone, with a Hough transform voting scheme on the optical flow. With these parameters, we propose several candidate locations for the target. Thus, for every frame of the video sequence, our candidate set consists of object detector boxes as well as geometry estimation proposals. Note that state-of-the-art tracking-by-detection approaches are limited to detector proposals alone. The second contribution we make is in selecting the best proposal from the candidate set using multiple cues – objectness scores [203] computed with edge responses [37] and motion boundaries [184], and the detection score.

Related Work. Tracking-by-detection methods learn an initial discriminative model of the object from the first frame in the sequence (e.g., with a support vector machine (SVM) [65], [171]), evaluate it to detect the most likely object location in subsequent frames, and then update the object model with these new detections. To avoid the well-known problem of drift [107] in tracking, some methods combine detector- and tracker-based components [78], control the training set with learning [171], or maintain a pool of trackers [A12]. These methods however: (i) lack robustness to severe changes in object size [78] or scale [A12], or (ii) cannot handle commonly-occurring scenarios where the object is undergoing a geometric transformation [171], such as the rotation shown in Figure 3.8-bottom. A more complete discussion on related work is presented in [A13].

3.2.1 Proposal selection for tracking

The main components of our framework for online object tracking are: (i) learning the initial detector, (ii) building a rich candidate set of object locations in each frame, consisting of proposals from the detector as well as the estimated geometric transformations, (iii) evaluating all the proposals in each frame with multiple cues to select the best one, and (iv) updating the detector model. We now present all these components and then provide implementation details in [A13].

Initial detector. We learn the initial detector model with a training set consisting of one positive sample, available as a bounding box annotation in the first frame, and several negative bounding box samples which are automatically extracted from the entire image. We use HOG features [31], [49] computed for these bounding boxes and learn the detector with a linear SVM, similar to other tracking-by-detection approaches [A12],

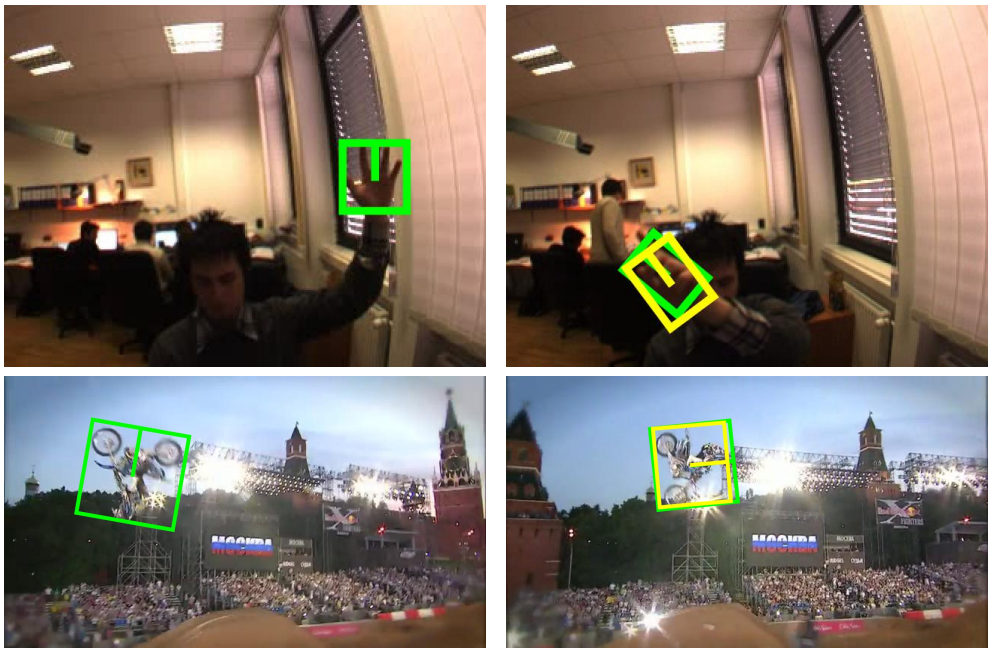


Figure 3.9 – Two sequences from the VOT dataset [105]. In each row, we show the ground truth (in green) in the first frame (left) and the best geometry proposal in a subsequent frame (in yellow in the right image). We show a vertical line on each box to visualize the rotation angle with respect to image edges.

[171]. The detector is then evaluated on subsequent frames to estimate the candidate locations of the object. Rather than make a suboptimal decision by choosing the best detection as the object location in a frame, we extract the top k detections and build a candidate set. We augment this set with proposals estimated from the transformations undergone by the object, as described in the following.

Estimating geometry and proposals. In the examples shown in Figure 3.9, the object of interest is undergoing a geometric transformation (rotation in these examples). None of the top detector proposals can capture the object accurately in this case. To address this issue, we explicitly estimate the transformation undergone by the object and then use it to enrich the candidate set with additional geometric proposals.

We represent the geometric transformation with a similarity matrix [66]. Note that other transformations like homography can also be used. The similarity transformation is defined by four parameters – one each for rotation and scale, and two for translation. In this work, we estimate them with a Hough transform voting scheme using frame-to-frame optical flow correspondences. We begin by computing the optical flow between frames $t-1$ and t with a state-of-the-art flow estimation method [24]. The pixels within the object bounding box with flow values in frame $t-1$ then give us corresponding pixels in frame t. With a pair of these matches, given by two pixels in $t-1$ which are sufficiently distant from

each other, we estimate the four scalar parameters in the similarity matrix [66]. Every choice of a pair of corresponding matches gives us an estimate of the four parameters. We then use the Hough transform [72], wherein each pair of point-matches votes for a (4D) parameter set, to find the top k consistent parameter sets. This voting scheme allows us to filter out the incorrect correspondences due to common errors in optical flow estimation.

To sum up, we estimate the k most likely geometric transformations undergone by the object as the parameters with the top k votes. Each one of these transformations results in a candidate location of the object in t, by applying the corresponding similarity matrix to the object bounding box in $t-1$. Before adding all these k proposals to the candidate set, we perform an additional filtering step to ensure further robustness. To this end, we discard all the proposals: (i) which have a low confidence, given by the number of votes, normalized by the total number of point-matches, and (ii) those where the estimated angle is significantly different from estimations in earlier frames. We determine both these threshold values with Kalman filtering [79]. A threshold τ_t, used for filtering proposals in frame $t+1$, is given by:

$$\tau_t = \alpha\tau_{t-1} + k_t(d_t - \alpha\beta\tau_{t-1}), \tag{3.7}$$

$$k_t = (p_{t-1} + q)/(p_{t-1} + q + r), \tag{3.8}$$

$$p_t = (1 - k_t)(p_{t-1} + q), \tag{3.9}$$

where α, β, q, r are scalar parameters set empirically (see [A13]), k_t is the Kalman gain, p_t is the error estimate. The filter is initialized with $\tau_0 = 0$, $p_0 = 0.1$. Here, d_t is either the confidence score or the estimated angle of the selected proposal in frame t to determine the respective threshold.

Selecting proposals. At the end of the geometry estimation step, we have k proposals from the detector and (at most) k proposals from the similarity transformation in frame t. Now, the task is to find the *best* proposal most likely to contain the object in this frame. We use three cues, detection confidence score, objectness measures computed with object edges and motion boundaries, for this task. We first use the (normalized) detection confidence score computed for each proposal box with the SVM learned from the object annotation in the first frame and updated during tracking. This provides information directly relevant to the object of interest in a given sequence.

In cases where the detection scores of all the candidate boxes are statistically similar, we use cues extracted from object edges [37] and motion boundaries [184] in the image, referred to as edgeness scores. In other words, when the detection scores are inconclusive to choose the best proposal, we rely on edgeness measure. Here, all the top candidates contain the object of interest to some extent, and the task is to choose the one that best contains the object—a scenario well-suited for edgeness cue because a proposal box which accurately localizes the entire object has a stronger edge response compared to a box which overlaps partially with the object, thus resulting in a weaker edge score. The edgeness score is given by the number of edges (or motion boundaries) within a proposal box after discarding contours that intersect with the box's boundary, as in [203]. We

Figure 3.10 – Two examples of edge and motion boundary responses. In each row, from left to right, we show the original image (with the object in a green box), edges and motion boundaries. The edges provide a stronger response in the example in the first row while motion boundaries are more effective in the example in the second row.

compute edgeness scores with edges and motion boundaries separately and use the one with a stronger response. As shown in the examples in Figure 3.10, these two cues are complementary. Then, the proposal with the best edgeness score is selected.

We follow this approach, rather than using a combined score, e.g., a weighted combination of SVM and edgeness scores, because setting this weight manually is suboptimal, and learning it is not possible due to the lack of standard training-validation-test datasets for tracking. In contrast, our approach uses object-specific SVM and generic edgeness scores effectively, and circumvents the need for a manually set weight to combine scores.

Updating the detector. Having computed the best proposal containing the object, box_t, we use it as a positive exemplar to learn a new object model. In practice, we update the current model incrementally with this new sample; see [A13]. This new detector is then evaluated in frame $t + 1$, to continue tracking the object.

3.2.2 Results: Visual object tracking

We now present our empirical evaluation on the visual object tracking (VOT) benchmark dataset, and compare with several recent methods. The results of all the methods we compare with are directly taken from the VOT toolkit. Additional results are provided in [A13].

No.	Method	Acc.	Robust.	Avg.
1	Our-ms-rot	**6.07**	**8.58**	**7.33**
2	Our-ms	**4.73**	10.13	**7.43**
3	DSST [32]	6.78	13.99	**10.39**
4	SAMF	6.46	15.65	11.06
5	DGT	12.67	10.13	11.4
6	KCF [68]	**6.16**	16.71	11.44
7	PLT_{14}	16.04	**6.98**	11.51
8	PLT_{13}	19.74	**4.00**	11.87
9	eASMS	15.37	15.1	15.24
10	Our-ss	16.11	14.47	15.29
11	ACAT	15.1	16.78	15.94
12	MatFlow	23.82	9.67	16.74
13	HMMTxD	11.08	22.51	16.8
14	MCT	18.47	15.14	16.81
15	qwsEDFT	19.06	21.15	20.11
16	ACT	22.68	18.1	20.39
17	ABS	22.34	20.49	21.42
18	VTDMG	23.22	19.94	21.58
19	LGTv1	31.05	12.68	21.87
20	BDF	24.92	19.39	22.15

Table 3.1 – Performance of the top 20 methods on the VOT2014 dataset. We show the ranking on accuracy (Acc.) and robustness (Robust.) measures, as well as the overall rank (Avg.). The top performer in each measure is shown in red, and the second and third best are in blue and green respectively. We show three variants of our method – Our-ms-rot: uses multiscale detector and geometry proposals, Our-ms: uses only multiscale detector proposals, and Our-ss: uses only single-scale detector proposals.

The results in Table 3.1 correspond to the baseline experiment in the toolkit, where each tracker is executed 15 times with the same ground truth annotation in the first frame, and the overall rank is computed as described in [A13]. Overall, our proposal selection method using detector and geometry proposals ("Our-ms-rot" in the table) performs significantly better than DSST [32], the winner of the VOT2014 challenge, with an average rank of 7.33 vs 10.39. We also evaluated two variants of our approach. The first one "Our-ms" uses only the multiscale detector proposals and the second variant "Our-ss" is limited to proposals from the detector evaluated only at a single scale. The overall rank of these two variants is lower than our full method: 7.43 and 15.29 respectively. The variant based on multiscale detector proposals performs better on the accuracy measure compared to the full method, but is significantly worse on the robustness measure. In other words, this variant fails on many more frames than the full method, and benefits from the resulting reinitializations. Due to significant changes in scale that occur in several frames in the dataset, the variant based on a single-scale detector performs rather

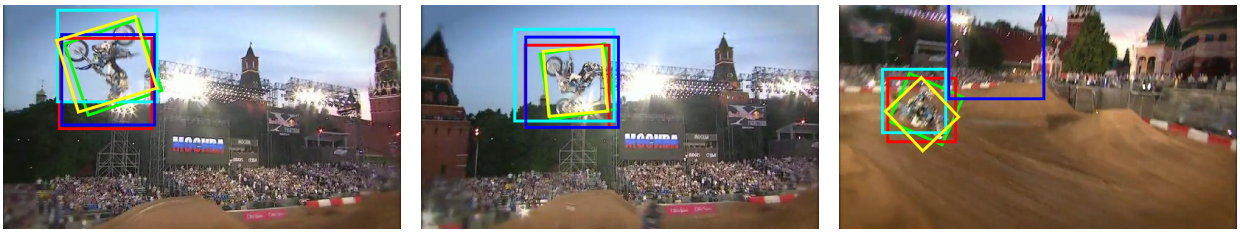


Figure 3.11 – Sample result on the *motocross* sequence. We compare with several state-of-the-art trackers. Ground truth: green, Our result: yellow, DSST [32]: red, PLT_{14}: cyan, Struck [65]: blue. (*Best viewed in colour.*)

poorly. CMT [116], the key point based tracker which estimates rotation and scale of the object, has an average rank of 24.43 and is ranked 28th on the list. With the evaluation protocol of reinitializing the tracker when it fails, we found that using edgeness measure did not show a significant difference compared to the detection score. Thus, to keep the comparison with respect to the large number of trackers (38) reasonable, we show a subset of variants of our methods on VOT.

Our tracker is ranked third on the robustness measure. PLT_{14} and PLT_{13}, which are two variants of *Struck* [65], are first and second respectively. PLT_{13}, the winner of the VOT2013 challenge, uses object-background colour histograms to weight the features within the bounding box during SVM training. In essence, it attempts to refine the object representation from an entire bounding box to an object segmentation. PLT_{14} is the multiscale version of PLT_{13}, and it shows better accuracy at the cost of a lower robustness. The segmentation cue used in these methods makes them accurate because it provides a more precise object representation than a bounding box. However, in many situations, e.g., fast motion, lack of strong object-background colour priors, it is likely to fail. This is evident from the significantly lower accuracy of the PLT methods – 16.04, 19.74 compared to 6.07 of our approach (see Table 3.1).

We present a sample qualitative result in Figure 3.11. The *motocross* sequence is considered as one of the most challenging sequences in this benchmark, based on the number of trackers that fail on it. Our approach performs significantly better other methods. We compare to the results of DSST and PLT_{14}, which performed well on the VOT2014 challenge, and also *Struck*, the top performer on several datasets.

3.2.3 Summary

This work presents a new tracking-by-detection framework for online object tracking. Our approach begins with building a candidate set of object location proposals extracted using a learned detector model. It is then augmented with novel proposals computed by estimating the geometric transformation(s) undergone by the object. We localize the object by selecting the best proposal from this candidate set using multiple cues: detection confidence score, edges and motion boundaries.

3.3 Learning Motion Features for Segmentation

Video object segmentation is the task of extracting spatio-temporal regions that correspond to object(s) moving in at least one frame in the video sequence. The top-performing methods for this problem [45], [127] continue to rely on hand-crafted features and do not leverage a learned video representation, despite the impressive results achieved by CNNs for other vision tasks, e.g., image segmentation [133], object detection [142]. Very recently, there have been attempts to build CNNs for video object segmentation [25], [76], [84]. Yet, they suffer from various drawbacks. For example, [25], [84] rely on a manually-segmented subset of frames (typically the first frame of the video sequence) to guide the segmentation pipeline. The approach of [76] does not require manual annotations, but remains frame-based, failing to exploit temporal consistency in videos. Furthermore, none of these methods has a mechanism to *memorize* relevant features of objects in a scene. In this work, we propose a novel framework to address these issues.

We present a two-stream network with an explicit memory module for video object segmentation (see Figure 3.12). The memory module is a convolutional gated recurrent unit (GRU) that encodes the spatio-temporal evolution of object(s) in the input video sequence. This spatio-temporal representation used in the memory module is extracted from two streams—the appearance stream which describes static features of objects in the video, and the temporal stream which captures the independent object motion.

The temporal stream separates independent object and camera motion with our motion pattern network (MP-Net), a trainable model, which takes optical flow as input and outputs a per-pixel score for moving objects. Inspired by fully convolutional networks (FCNs) [38], [103], [148], we propose a related encoder-decoder style architecture to accomplish this two-label classification task. The network is trained from scratch with synthetic data [108]. Pixel-level ground-truth labels for training are generated automatically (see Figure 3.13(d)), and denote whether each pixel has moved in the scene. The input to the network is flow fields, such as the one shown in Figure 3.13(c). More details of the network, and the training procedure are provided in Section 3.3.2. With this training, our model learns to distinguish motion patterns of objects and background.

The appearance stream is the DeepLab network [27], [28], pretrained on the PASCAL VOC segmentation dataset, and it operates on individual video frames. With the spatial and temporal CNN features, we train the convolutional GRU component of the framework to learn a *visual memory* representation of object(s) in the scene. Given a frame t from the video sequence as input, the network extracts its spatio-temporal features and: (i) computes the segmentation using the memory representation aggregated from all frames previously seen in the video, (ii) updates the memory unit with features from t. The segmentation is improved further by processing the video in the forward and the backward directions in the memory unit, with our *bidirectional convolutional GRU*.

Related Work. Our work is related to: motion and scene flow estimation, video object segmentation, and recurrent neural networks. We discuss a selection of these works here, and provide a more elaborate discussion in [A30]. In concurrent work, Jain et al. [76] presented a deep network to segment independent motion in the flow field. While

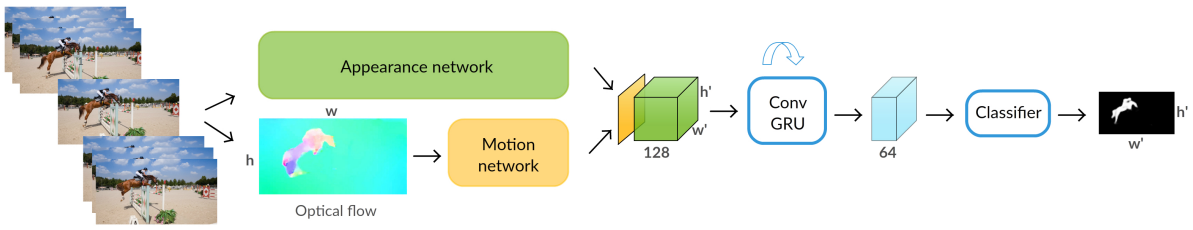


Figure 3.12 – Overview of our segmentation approach. Each video frame is processed by the appearance (green) and the motion (yellow) networks to produce an intermediate two-stream representation. The ConvGRU module combines this with the learned visual memory to compute the final segmentation result. The width (w') and height (h') of the feature map and the output are w/8 and h/8 respectively.

(a) (b) (c) (d)

Figure 3.13 – (a,b) Two example frames from a sequence in the FlyingThings3D dataset [108]. The camera is in motion in this scene, along with four independently moving objects. (c) Ground-truth optical flow of (a), which illustrates motion of both foreground objects and background with respect to the next frame (b). (d) Ground-truth segmentation of moving objects in this scene.

their approach is related to ours, they use frame pairs from real videos, in contrast to synthetic data in our case. Since obtaining accurate ground truth moving object segmentation labels is prohibitively expensive for a large dataset, they rely on an automatic, heuristic-based label estimation approach, which results in noisy annotations. Our approach to solve the segmentation problem does not require manually-marked regions. Several methods in this paradigm generate an over-segmentation of videos [22], [62], [83]. While this can be a useful intermediate step for some recognition tasks in video, it has no notion of objects.

3.3.1 Learning to segment moving objects in videos

We start by describing the overall architecture of our video object segmentation framework. It takes video frames together with their estimated optical flow as input, and outputs binary segmentations of moving objects, as shown in Figure 3.12. We target the most general form of this task, wherein objects are to be segmented in the entire video if they move in at least one frame. The proposed model is comprised of three key

components: appearance and motion networks, and a visual memory module described below.

Appearance network. The purpose of the appearance stream is to produce a high-level encoding of a frame that will later aid the visual memory module in forming a representation of the moving object. It takes a w × h RGB frame as input and produces a 128 × w/8 × h/8 feature representation (shown in green in Figure 3.12), which encodes the semantic content of the scene. As a baseline for this stream we use the large-FOV, VGG16-based version of the DeepLab network [27]. This network's architecture is based on dilated convolutions [27], which preserve a relatively high spatial resolution of features, and also incorporate context information in each pixel's representation. It is pretrained on a semantic segmentation dataset [43], resulting in features that can distinguish objects from background as well as from each other—a crucial aspect for the video object segmentation task. We also experiment with upgrading the appearance stream to DeepLab-v2 [28], a more recent version of the model, where the VGG16 architecture is replaced with ResNet101, and the network is additionally pretrained on the COCO semantic segmentation dataset [101].

Motion network. For the temporal stream we employ a CNN pretrained for the motion segmentation task. It is trained to estimate independently moving objects (i.e., irrespective of camera motion) based on optical flow computed from a pair of frames as input; see Section 3.3.2 for details. This stream (shown in yellow in Figure 3.12) produces a w/4 × h/4 motion prediction output, where each value represents the likelihood of the corresponding pixel being in motion. Its output is further downsampled by a factor 2 (in w and h) to match the dimension of the appearance stream output.

The intuition behind using two streams is to benefit from their complementarity for building a strong representation of objects that evolves over time. For example, both appearance and motion networks are equally effective when an object is moving in the scene, but as soon as it becomes stationary, the motion network can not estimate the object, unlike the appearance network. We leverage this complementary nature, as done by two-stream networks for other vision tasks [164]. Note that our approach is not specific to the particular networks described above, but is in fact a general framework for video object segmentation. As shown in [A30], its components can easily replaced with other networks, providing scope for future improvement.

Memory module. The third component, i.e., a visual memory module, takes the concatenation of appearance and motion stream outputs as its input. It refines the initial estimates from these two networks, and also memorizes the appearance and location of objects in motion to segment them in frames where: (i) they are static, or (ii) motion prediction fails. The output of this ConvGRU memory module is a 64 × w/8 × h/8 feature map obtained by combining the two-stream input with the internal state of the memory module, as described in detail in Section 3.3.3. We further improve the model by processing the video bidirectionally. The output from the ConvGRU module is processed

by a 1×1 convolutional layer and a softmax nonlinearity to produce the final pairwise segmentation result.

3.3.2 Motion pattern network

Our MP-Net takes the optical flow field corresponding to two consecutive frames of a video sequence as input, and produces per-pixel motion labels. We treat each video as a sequence of frame pairs, and compute the labels independently for each pair. The network comprises several "encoding" (convolutional and max-pooling) and "decoding" (upsampling and convolutional) layers. The motion labels are produced by the last layer of the network, which are then rescaled to the original image resolution. We train the network on synthetic data—a scenario where ground-truth motion labels can be acquired easily.

Network architecture. Our encoder-decoder style network is motivated by the goal of segmenting diverse motion patterns in flow fields, which requires a large receptive field as well as an output at the original image resolution. A large receptive field is critical to incorporate context into the model. For example, when the spatial region of support (for performing convolution) provided by a small receptive field falls entirely within an object with non-zero flow values, it is impossible to determine whether it is due to object or camera motion. On the other hand, a larger receptive field will include regions corresponding to the object as well as background, providing sufficient context to determine what is moving in the scene. The second requirement of output generated at the original image resolution is to capture fine details of objects, e.g., when only a part of the object is moving. Our network satisfies these two requirements with: (i) the encoder part learning features with receptive fields of increasing sizes, and (ii) the decoder part upsampling the intermediate layer outputs to finally predict labels at the full resolution. This approach is inspired by recent advances in semantic segmentation, where similar requirements are encountered [148]. Implementation details of the architecture are provided in [A30].

Training with synthetic data. We need a large number of fully-labelled examples to train a convolutional network such as the one we propose. In our case, this data corresponds to videos of several types of objects, captured under different conditions (e.g., moving or still camera), with their respective moving object annotations. No large dataset of real-world scenes satisfying these requirements is currently available, predominantly due to the cost of generating ground-truth annotations and flow for every frame. We adopt the popular approach of using synthetic datasets, followed in other work [38], [108]. Specifically, we use the FlyingThings3D dataset [108] containing 2250 video sequences of several objects in motion, with ground-truth optical flow. We augment this dataset with ground-truth moving object labels, which are accurately estimated using the disparity values and camera parameters available in the dataset. See Figure 3.13(d) for an illustration.

(a) (b) (c)

Figure 3.14 – Each row shows: (a) example frame from a sequence in FlyingThings3D, (b) ground-truth optical flow of (a), which illustrates motion of both foreground objects and background, with respect to the next frame, and (c) our estimate of moving objects in this scene with ground-truth optical flow as input.

We train the network with mini-batch SGD under several settings. The one trained with ground-truth optical flow as input shows the best performance. Note that, while we use ground-truth flow for training and evaluating the network on synthetic datasets, all our results on real-world test data use only the estimated optical flow. After convergence of the training procedure, we obtain a learned model for motion patterns.

Our approach capitalizes on the recent success of CNNs for pixel-level labeling tasks, such as semantic image segmentation, which learn feature representations at multiple scales in the RGB space. The key to their top performance is the ability to capture local patterns in images. Various types of object and camera motions also produce consistent local patterns in the flow field, which our model is able to learn to recognize. This gives us a clear advantage over other pixel-level motion estimation techniques [16], [115] that can not detect local patterns. Motion boundary based heuristics used in [127] can be seen as one particular type of pattern, representing independent object motion. Our model is able to learn many such patterns, which greatly improves the quality and robustness of motion estimation.

Detecting motion patterns. We apply our trained model on synthetic (FlyingThings3D) as well as real-world (DAVIS, FBMS, SegTrack-v2) test data. Figure 3.14 shows sample predictions of our model on the FlyingThings3D test set with ground-truth optical flow as input. Examples in the first two rows show that our model accurately identifies fine details in objects: thin structures even when they move slightly, such as the neck of the guitar in the top-right corner in the first row (see the subtle motion in the opti-

(a) (b) (c)

Figure 3.15 – Sample results on the DAVIS dataset for MP-Net. Each row shows: (a) video frame, (b) optical flow estimated with LDOF [24], (c) output of our MP-Net with LDOF flow as input.

cal flow field (b)), fine structures like leaves in the vase, and the guitar's headstock in the second row. Furthermore, our method successfully handles objects exhibiting highly varying motions in the second example. The third row shows a limiting case, where the receptive field of our network falls entirely within the interior of a large object, as the moving object dominates. Traditional approaches, such as RANSAC, do not work in this case either.

In order to detect motion patterns in real-world videos, we first compute optical flow with popular methods [73], [144], [170]. With this flow as input to the network, we estimate a motion label map, as shown in the examples in Figure 3.15(c). Although the prediction of our frame-pair feedforward model is accurate in several regions in the frame ((c) in the figure), we are faced with two challenges, which were not observed in the synthetic training set. The first one is motion of *stuff* [5] in a scene, e.g., patterns on the water due to the kiteboarder's motion (first row in the figure), which is irrelevant for moving object segmentation. The second one is significant errors in optical flow, e.g., in front of the pram ((b) in the bottom row in the figure). Furthermore, this motion segmentation approach is purely frame-based, thus unable to exploit temporal consistency in a video, and does not segment object in frames where they stop moving. In our previous work [A27] we introduced post-processing steps to handle some of these problems. In particular, we incorporated an objectness map computed with object proposals [133] to suppress motion corresponding to stuff, as well as false positives due to errors in flow estimation. This post-processing allowed the method to achieve competitive results, but it remained frame-level. The video object segmentation framework presented in this work addresses all these issues, as shown experimentally.

3.3.3 ConvGRU visual memory module

The key component of the ConvGRU module is the state matrix h, which encodes the visual memory. For frame t in the video sequence, ConvGRU uses the two-stream representation x_t and the previous state h_{t-1} to compute the new state h_t. The dynamics of this computation are guided by an update gate z_t, a forget gate r_t. The states and the gates are 3D tensors, and can characterize spatio-temporal patterns in the video, effectively memorizing which objects move, and where they move to. These components are computed with convolutional operators and nonlinearities as follows.

$$z_t = \sigma(x_t * w_{xz} + h_{t-1} * w_{hz} + b_z), \tag{3.10}$$
$$r_t = \sigma(x_t * w_{xr} + h_{t-1} * w_{hr} + b_r), \tag{3.11}$$
$$\tilde{h}_t = \tanh(x_t * w_{x\tilde{h}} + r_t \odot h_{t-1} * w_{h\tilde{h}} + b_{\tilde{h}}), \tag{3.12}$$
$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t, \tag{3.13}$$

where $\odot$ denotes element-wise multiplication, $*$ represents a convolutional operation, σ is the sigmoid function, w's are learned transformations, and b's are bias terms.

The new state h_t in (3.13) is a weighted combination of the previous state h_{t-1} and the candidate memory $\tilde{h}_t$. The update gate z_t determines how much of this memory is incorporated into the new state. If z_t is close to zero, the memory represented by $\tilde{h}_t$ is ignored. The reset gate r_t controls the influence of the previous state h_{t-1} on the candidate memory $\tilde{h}_t$ in (3.12), i.e., how much of the previous state is let through into the candidate memory. If r_t is close to zero, the unit forgets its previously computed state h_{t-1}.

The gates and the candidate memory are computed with convolutional operations over x_t and h_{t-1} shown in equations (3.10-3.12). We illustrate the computation of the candidate memory state $\tilde{h}_t$ in Figure 3.16. The state at $t-1$, h_{t-1}, is first multiplied (element-wise) with the reset gate r_t. This modulated state representation and the input x_t are then convolved with learned transformations, $w_{h\tilde{h}}$ and $w_{x\tilde{h}}$ respectively, summed together with a bias term $b_{\tilde{h}}$, and passed though a tanh nonlinearity. In other words, the visual memory representation of a pixel is determined not only by the input and the previous state at that pixel, but also its local neighbourhood. Increasing the size of the convolutional kernels allows the model to handle spatio-temporal patterns with larger motion.

The update and reset gates, z_t and r_t, are computed in an analogous fashion using a sigmoid function instead of tanh. Our ConvGRU applies a total of six convolutional operations at each time step. All the operations detailed here are fully differentiable, and thus the parameters of the convolutions (w's and b's) can be learned in an end-to-end fashion with back propagation through time [189]. In summary, the model learns to combine appearance features of the current frame with the memorized video representation to refine motion predictions, or even fully restore them from the previous observations in case a moving object becomes stationary.

Bidirectional processing. Consider an example where an object is stationary at the beginning of a video sequence, and starts to move in the latter frames. Our approach

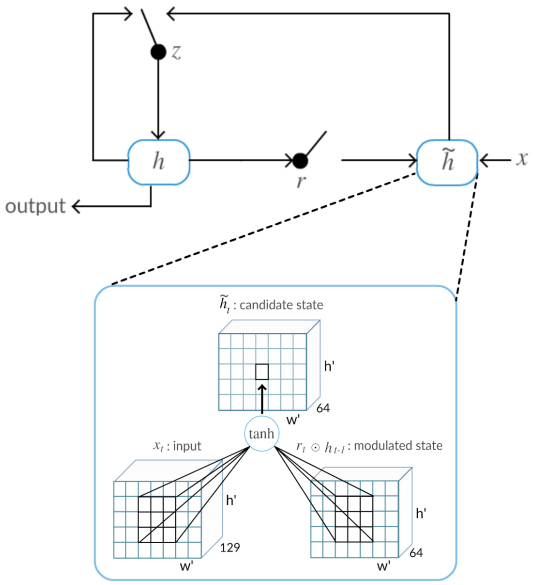


Figure 3.16 – Illustration of ConvGRU with details for the candidate hidden state module, where $\tilde{h}_t$ is computed with two convolutional operations and a tanh nonlinearity.

described so far, which processes video frames sequentially (in the forward direction), starting with the first frame can not segment the object in the initial frames. This is due to the lack of prior memory representation of the object in the first frame. We improve our framework with a bidirectional processing step, inspired by the application of recurrent models bidirectionally in the speech domain [60].

The bidirectional variant of our ConvGRU is composed of two ConvGRU instances with identical learned weights, which are run in parallel. The first one processes frames in the forward direction, starting with the first frame. The second instance process frames in the backward direction, starting with the last video frame. The activations from these two directions are concatenated at each time step, and then passed through a convolutional layer to finally produce a $64 \times w/8 \times h/8$ output for each frame. Pixel-wise segmentation from this activation is the result of the last convolutional layer and softmax nonlinearity, as in the unidirectional case. Bidirectional ConvGRU is used both in training and in testing, allowing the model to learn to aggregate information over the entire video. In addition to handling cases where objects move in the latter frames, it improves the ability of the model to correct motion prediction errors.

Training. We train our visual memory module with the back propagation through time algorithm [189], which unrolls the recurrent network for n time steps and keeps all the intermediate activations to compute the gradients. Thus, our ConvGRU model, which has six internal convolutional layers, trained on a video sequence of length n, is equivalent to a $6n$ layer CNN for the unidirectional variant, or $12n$ for the bidirectional model at training time. This memory requirement makes it infeasible to train the whole model,

Measure		KEY [99]	MSG [23]	NLC [45]	CUT [82]	FST [127]	MP-Net-F [A27]	FSG [76]	ARP [86]	Ours
$\mathcal{J}$	Mean	49.8	53.3	55.1	55.2	55.8	70.0	70.7	76.2	78.2
	Recall	59.1	61.6	55.8	57.5	64.9	85.0	83.5	91.1	89.1
	Decay	14.1	2.4	12.6	2.3	0.0	1.4	1.5	7.0	4.1
$\mathcal{F}$	Mean	42.7	50.8	52.3	55.2	51.1	65.9	65.3	70.6	75.9
	Recall	37.5	60.0	51.9	61.0	51.6	79.2	73.8	83.5	84.7
	Decay	10.6	5.1	11.4	3.4	2.9	2.5	1.8	7.9	3.5
$\mathcal{T}$	Mean	25.2	29.1	41.4	26.3	34.3	56.3	32.8	39.3	20.2

Table 3.2 – Comparison to recent methods on DAVIS with intersection over union ($\mathcal{J}$), F-measure ($\mathcal{F}$), and temporal stability ($\mathcal{T}$).

including appearance and motion streams, end-to-end. We resort to using pretrained versions of the appearance and motion networks, and train the ConvGRU.

In contrast to our motion segmentation model, which is learned on synthetic videos, we use the training split of the DAVIS dataset [132] for learning the ConvGRU weights. Despite being an order of magnitude smaller, DAVIS consists of realistic videos, which turns out to be crucial for effective use of appearance stream to correct motion estimation errors. Since objects move in all the frames in DAVIS, it biases the memory module towards the presence of an uninterrupted motion stream. This results in the ConvGRU learned from this data failing, when an object stops to move in a test sequence. We augment the training data to simulate such *stop-and-go* scenarios to learn a more robust model for realistic videos. To this end, we create additional sequences where we duplicate the last frame five times, i.e., we create a part of the video in which the object is static. This setting allows ConvGRU to learn how to segment objects even if they are static. It explicitly memorizes the moving object in the initial part of the sequence, and then segments it in frames where the motion stops. We also augment the training data by duplicating the first five frames to simulates scenarios where the object is static in the beginning of a sequence.

3.3.4 Results: Video segmentation

We present the performance of our complete method: DeepLab-v2 appearance stream, ConvGRU memory module trained on DAVIS, Bi-directional processing, MP-Net fine-tuned on FusionSeg with FlowNet 2.0 used or flow estimation (FSeg + FNet) and Dense-CRF [92] post-processing, on the DAVIS validation set. Other analyses are available in [A30], along with the implementation details.

Table 3.2 compares our approach to recent methods on DAVIS. In addition to comparing our results to the top-performing unsupervised approaches reported in [132], we included the results of recent methods from the benchmark website: [2] CUT [82], FSG [76]

2. http://davischallenge.org/soa_compare.html

Figure 3.17 – Qualitative comparison with the top-performing methods on DAVIS. Left to right: ground truth, results of MP-Net-F [A27], FSG [76], ARP [86], and our method.

and ARP [86], as well as the frame-level variant of our method: MP-Net-F [A27]. This frame-level approach augments our motion estimation model with an heuristic-based objectness score and uses DenseCRF for postprocessing (boundary refinement). Our method outperforms ARP [86], the previous state of the art by 2% on the mean IoU measure. We also observe an 8.2% improvement over MP-Net-F in mean IoU and 36.1% in temporal stability, which clearly demonstrates the significance of the visual memory module.

Figure 3.17 shows qualitative results of our approach, and the next three top-performing methods on DAVIS: MP-Net-F [A27], FSG [76] and ARP [86]. In the first row, our method fully segments the dancer, whereas MP-Net-F and FSG miss various parts of the person and ARP segments some of the people in the background. All these approaches use heuristics to combine motion and appearance cues, which become unreliable in cluttered scenes with many objects. Our approach does not include any heuristics, which makes it robust to this type of errors. In the second row, all the methods segment the car, but only our approach does not leak into other cars in the video, showing high discriminability. In the next row, our approach is able to fully segment a complex object, whereas the other methods either miss parts of it (MP-Net-F and FSG) or segment background regions as moving (ARP). In the last row, we illustrate a failure case of our method. The people in the background move in some of the frames in this example. MP-Net-F, FSG and our method segment them to varying extents. ARP focuses on the foreground object, but misses a part of it.

3.3.5 Summary

Our method for video object segmentation combines two complementary sources of information: appearance and motion, with a visual memory module, realized as a bidirectional convolutional gated recurrent unit. To separate object motion from camera motion we introduce a CNN-based model, which is trained using synthetic data to segment independently moving objects in a flow field. The ConvGRU module encodes spatio-temporal evolution of objects in a video based on a state-of-the-art appearance representation, and uses this encoding to improve motion segmentation.

Chapter 4

Learning for Recognition

This chapter is based on the following publications.

— F. M. Castro, M. J. Marín-Jiménez, N. Guil, C. Schmid, and K. Alahari, "End-to-End Incremental Learning," ECCV, 2018 [PDF]

— K. Shmelkov, C. Schmid, and K. Alahari, "Incremental Learning of Object Detectors without Catastrophic Forgetting," ICCV, 2017 [PDF]

— P. Tokmakov, K. Alahari, and C. Schmid, "Weakly-Supervised Semantic Segmentation using Motion Cues," ECCV, 2016 [PDF]

This chapter focuses on two learning paradigms for computer vision problems. The first one is on weakly-supervised learning for semantic segmentation (Section 4.1). We present a learning approach in the context of a fully-convolutional neural architecture, which only uses annotation labels at the image level instead of the expensive pixelwise segmentation labels. The other body of work considers the incremental learning problem (Section 4.2), where an initial model trained on a set of classes is adapted to additionally recognize objects of new classes, in the absence of all the original training data.

4.1 Weakly-Supervised Semantic Segmentation

Weakly-supervised learning for semantic segmentation is particularly important, as acquiring a training set by labeling images manually at the pixel level is significantly more expensive than assigning class labels at the image level. Recent segmentation approaches have used weak annotations in several forms: bounding boxes around objects [111], [191], image labels denoting the presence of a category [134], [177] or a combination of the two [126]. All these previous approaches only use annotation in images, i.e., bounding boxes, image tags, as a weak form of supervision. Naturally, additional cues would come in handy to address this challenging problem. As noted in [23], motion is one such cue for semantic segmentation, which helps us identify the extent of objects and their boundaries in the scene more accurately. It is however not leveraged for weakly-supervised semantic segmentation. In this work, we aim to fill this gap by learning an accurate segmentation model with the help of motion cues extracted from weakly-annotated videos.

Our proposed framework is based on fully convolutional neural networks (FCNNs) [27], [46], [103], [202], which extend deep CNNs, and are able to classify every pixel in an input image in a single forward pass of the network. While FCNNs show state-of-the-art results on segmentation benchmark datasets, they require thousands of pixel-level annotated images to train on—a requirement that limits their utility. There have been some attempts [126], [129], [130], [134] to train FCNNs with weakly-annotated images, but they remain inferior in performance to their fully-supervised equivalents (see Figure 4.1). In this work, we develop a new CNN variant named M-CNN, which leverages the motion cues in weakly-labeled videos, in the form of unsupervised motion segmentation, e.g., [127]. It builds on the architecture of FCNN by adding a motion segmentation based label inference step, as shown in Figure 4.2. In other words, the predictions from the FCNN layers and motion segmentation jointly determine the loss used to learn the network (see Section 4.1.1).

Our approach uses unsupervised motion segmentation from real-world videos, such as the YouTube-Objects [137] and the ImageNet-VID [1] datasets, to train the network. In this context, we are confronted with two main challenges. The first one is that even the best-performing algorithms cannot produce good motion segmentations consistently, and the second one is the ambiguity of video-level annotations, which cannot guarantee the presence of object in all the frames. We develop a novel scheme to address these challenges automatically without any manual annotations, apart from the labels assigned

1. http://vision.cs.unc.edu/ilsvrc2015/download-videos-3j16.php#vid

Image DeepLab [27] (full) EM-Adapt [126] (weak) M-CNN (weak)

Figure 4.1 – Comparison of fully [27] and weakly [126] supervised methods with our weakly-supervised M-CNN model.

at the video level, denoting the presence of objects somewhere in the video. To this end, we use motion segmentations as soft constraints in the learning process, and also fine-tune our network with a small number of video shots to refine it. Our trained model, despite using only 150 video labels, achieves performance similar to EM-Adapt [126] trained on more than 10,000 VOC image labels. Augmenting our training set with 1,000 VOC images results in a further gain, achieving the best performance on VOC 2012 test set in the weakly-supervised setting [A29].

Related work. Several segmentation methods learned from weak annotations have been proposed over the years: some of them use bounding boxes [111], [191], while others rely on image labels [177]. Traditional approaches for this task, such as [177], used a variety of hand-crafted visual features, namely, SIFT histograms, colour, texture, in combination with a graphical or a parametric structured model. Recent attempts [126], [129], [130], [134], which focus on adapting FCNNs for the weakly-supervised case use either a multiple instance learning (MIL) scheme [130], [134] or constraints on the distribution of pixel labels [126], [129] to define the loss function. Weakly-supervised FCNNs in [126], [129] define constraints on the predicted pixel labels. In summary, none of these approaches exploit cues extracted from video for segmentation. A more complete discussion on related work is available in [A29].

4.1.1 Learning semantic segmentation from video

We train our network by exploiting motion cues from video sequences. Specifically, we extract unsupervised motion segments from video, with algorithms such as [127], and use them in combination with the weak labels at the video level to learn the network. We sample frames from all the video sequences uniformly, and assign them the class label of the video. This collection forms our training dataset, along with their corresponding motion segments.

The parameters of M-CNN are updated with a standard mini-batch SGD, similar to other CNN approaches [126], with the gradient of a loss function. Here, the loss measures the discrepancy between the ground truth segmentation label and the label predicted at each pixel. Thus, in order to learn the network for the semantic segmentation task, we need pixel-level ground truth for all the training data. These pixel-level labels

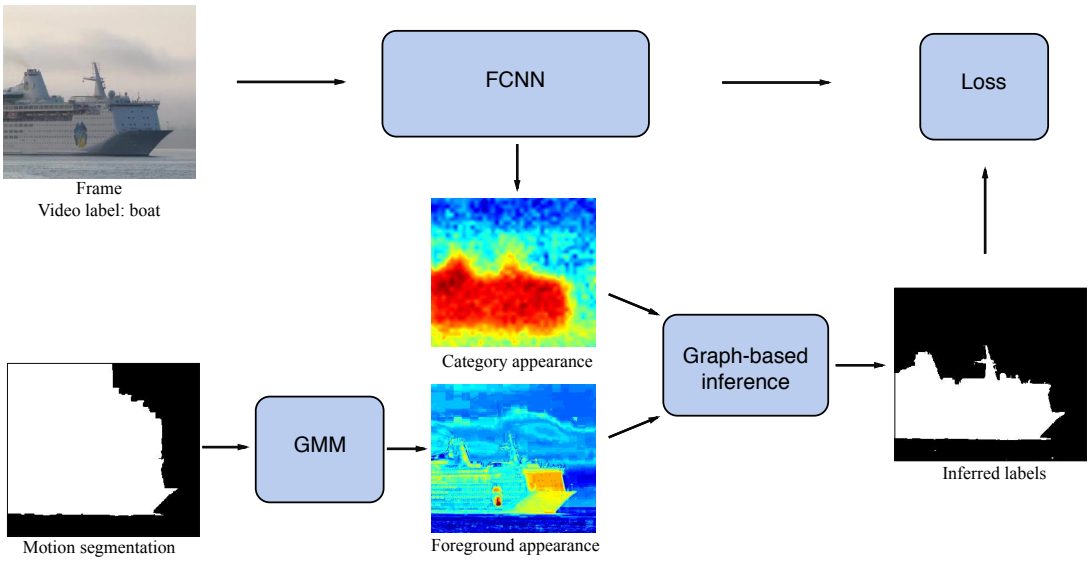


Figure 4.2 – Overview of our M-CNN framework for learning semantic segmentation, where we show only one frame from a video example for clarity. The soft potentials (foreground appearance) computed from motion segmentation and the FCNN predictions (category appearance) jointly determine the latent segmentation (inferred labels) to compute the loss, and thus the network update.

are naturally latent variables in the context of weakly-supervised learning. Now, the task is to estimate them for our weakly-labeled videos. An ideal scenario in this setting would be near-perfect motion segmentations, which can be directly used as object ground truth labels. However, in practice, not only are the segmentations far from perfect (see Figure 4.3), but also fail to capture moving objects in many of the shots. This makes a direct usage of motion segmentation results impossible. To address this, we propose a novel scheme, where motion segments are only used as soft constraints to estimate the latent variables together with object appearance cues.

The other challenges when dealing with real-world video datasets, such as YouTube-Objects and ImageNet-VID, are related to the nature of video data itself. On one hand, not all parts of a video contain the object of interest. For instance, a video from a show reviewing boats may contain shots with the host talking about the boat, and showing it from the inside for a significant part—content that is unsuitable for learning a segmentation model for the VOC 'boat' category. On the other hand, a long video can contain many nearly identical object examples which leads to an imbalance in the training set. We address both problems by fine-tuning our M-CNN with an automatically selected, small subset of the training data.

Network architecture. Our network is built on the DeepLab model for semantic image segmentation [27]. It is an FCNN, obtained by converting the fully-connected layers of the VGG-16 network [165] into convolutional layers. A few other changes are implemented to get a dense network output for an image at its full resolution efficiently.

Frame | Motion seg. [127] | Our label prediction

Figure 4.3 – Examples highlighting the importance of label prediction for handling imprecise motion segmentations (second column). The soft GMM potentials computed from motion segments together with network predictions produce better labels (third column) to learn the network. See Section 4.1.1 for details.

Our work builds on this network. We develop a more principled and effective label prediction scheme involving motion cues to estimate the latent variables, in contrast to the heuristic size constraints used in recent work [126] based on DeepLab.

Estimating latent variables with label prediction. Given an image of N pixels, let $\mathbf{p}$ denote the output of the softmax layer of the convolutional network. Then, $p_i^l \in [0, 1]$ is the prediction score of the network at pixel i for label l. The parameters of the network are updated with the gradient of the loss function, given by:

$$\mathcal{L}(\mathbf{x}, \mathbf{p}) = \sum_{i=1}^{N} \sum_{l=0}^{L} \delta(x_i - l) \log(p_i^l), \tag{4.1}$$

where $\mathbf{x}$ denotes ground truth segmentation labels in the fully-supervised case, $\mathbf{p}$ is the current network prediction, and $\delta(x_i - l)$ is the Dirac delta function, i.e., $\delta(x_i - l) = 1$, if $x_i = l$, and 0 otherwise. The segmentation label x_i of pixel i takes values from the label set $\mathbf{L} = \{0, 1, \ldots, L\}$, containing the background class (0) and L object categories. Naturally,

in the weakly-supervised case, ground truth segmentation labels are unavailable, and $\mathbf{x}$ represents latent segmentation variables, which need to be estimated. We perform this estimation with soft motion segmentation cues in this work.

Given the motion segmentation $\mathbf{s} = \{s_i | i = 1, \ldots, N\}$, where $s_i \in \{0, 1\}$ denotes whether a pixel i belongs to foreground (1) or background (0).[2] The regions assigned to foreground can represent multiple objects when the video is tagged with more than one object label. A simple way of transforming motion segmentation labels s_i into latent semantic segmentation labels x_i is with a hard assignment, i.e., $x_i = s_i$. This hard assignment is limited to videos containing a single object label, and also makes the assumption that motion segments are accurate and can be used as they are. We will see in our experiments that this performs poorly when using real-world video datasets (cf. 'M-CNN* hard' discussed in [A29]). We address this by using motion cues as soft constraints for estimating the label assignment $\mathbf{x}$ in the following.

Inference of the segmentation x. We compute the pixel-level segmentation $\mathbf{x}$ as the minimum of an energy function $E(\mathbf{x})$ defined by:

$$E(\mathbf{x}) = \sum_{i \in \mathcal{V}} \left(\psi_i^m(z_i) + \alpha \psi_i^{fc}(p_i^{x_i}) \right) + \sum_{(i,j) \in \mathcal{E}} \psi_{ij}(x_i, x_j), \tag{4.2}$$

where the set $\mathcal{V} = \{1, 2, \ldots, N\}$ represents all the pixels, z_i denotes the RGB colour at pixel i and the set $\mathcal{E}$ denotes all pairs of neighboring pixels in the image. Unary terms ψ_i^m and ψ_i^{fc} are computed from motion cues and current predictions of the neural network respectively with α being a scalar parameter, which weights their impact. The pairwise term ψ_{ij} imposes a smoothness over the label space.

The first unary term ψ_i^m captures the appearance of all foreground objects obtained from motion segments. To this end, we learn two Gaussian mixture models (GMMs), one each for foreground and background, with RGB values of pixel colours, similar to standard segmentation methods [127], [149]. The foreground GMM is learned with RGB values of all the pixels assigned to foreground in the motion segmentation. The background GMM is learned in a similar fashion with the corresponding background pixels. Given the RGB values of a pixel i, $\psi_i^m(z_i)$ is given by the negative log-likelihood of the corresponding GMM (background one for $l = 0$ and foreground otherwise). Using motion cues to generate this soft potential ψ_i^m helps us alleviate the issue of imperfect motion segmentation. The second unary term ψ_i^{fc} represents the learned object appearance model determined by the current network prediction $p_i^{x_i}$ for pixel i, i.e., $\psi_i^{fc}(p_i^{x_i}) = -\log(p_i^{x_i})$.

The pairwise term is based on a contrast-sensitive Potts model [20], [149] as:

$$\psi_{ij}(x_i, x_j) = \lambda(1 - \Delta(i, j))(1 - \delta(x_i - x_j)) \frac{\exp(-\gamma ||z_i - z_j||^2)}{\text{dist}(i, j)}, \tag{4.3}$$

where z_i and z_j are colours of pixels i and j, λ is a scalar parameter to balance the order of magnitude of the pairwise term with respect to the unary term, and γ is a scalar parameter set to 0.5 as in [127]. The function $\text{dist}(i, j)$ is the Euclidean distance

2. We do not include an index denoting the frame number in the video for brevity.

between pixels. The Dirac delta function $\delta(x_i - x_j)$ ensures that the pairwise cost is only applicable when two neighboring pixels take different labels. In addition to this, we introduce the term $(1 - \Delta(i, j))$, where $\Delta(i, j) = 1$ if pixels i and j both fall in the boundary region around the motion segment, and 0 otherwise. This accounts for the fact that motion segments may not always respect colour boundaries, and allows the minimization algorithm to assign different labels to neighboring pixels around motion edges.

We minimize the energy function (4.2) with an iterative GrabCut-like [149] approach, wherein we first apply the alpha expansion algorithm [21] to get a multi-label solution, use it to re-estimate the (background and foreground) GMMs, and then repeat the two steps for a few iterations. We highlight the importance of our label prediction technique with soft motion-cue constraints in Figure 4.3. Here, the original, binary motion predictions are imprecise (bottom two rows) or incorrect (top row) in all the examples, whereas using them as soft constraints in combination with the network prediction results in a more accurate estimation of the latent segmentation variables.

Fine-tuning M-CNN. We learn an initial M-CNN model from all the videos in the dataset which have sufficient motion information. To refine this model we add a fine-tuning step, which updates the parameters of the network with a small set of unique and reliable video examples. This set is built automatically by selecting one shot from each video sequence, whose motion segment has the highest overlap (intersection over union) score with the current M-CNN prediction. The intuition behind this selection criterion is that our M-CNN has already learned to discriminate categories of interest from the background, and thus, its predictions will have the highest overlap with precise motion segmentations. This model refinement leverages the most reliable exemplars and avoids near duplicates, often occurring within one video.

4.1.2 Results: With weakly-labeled image and video data

We trained our M-CNN in two settings. The first one is on purely video data, and the second on a combination of image and video data. We performed experiments primarily with the weakly-annotated videos in the YouTube-Objects v2.2 dataset [137]. Additionally, to demonstrate that our approach adapts to other datasets automatically, we used the ImageNet video (ImageNet-VID) dataset. In this section, we focus on results with the variant trained on image and video data. All the other experiments are discussed in [A29].

Figure 4.4 shows qualitative results of M-CNN (trained on VOC and YouTube-Objects) on a few sample images. These have much more accurate object boundaries than the best variant of EM-Adapt [126], which tends to localize the object well, but produces a 'blob-like' segmentation, see the last four rows in the figure in particular. The first three rows show example images containing multiple object categories. M-CNN recognizes the object classes more accurately, e.g., cow in row 5, than EM-Adapt, which confuses cow (shown in green) with horse (shown in magenta). Furthermore, our segmentation results compare favorably with the fully-supervised DeepLab [27] approach (see rows 4-6), highlighting

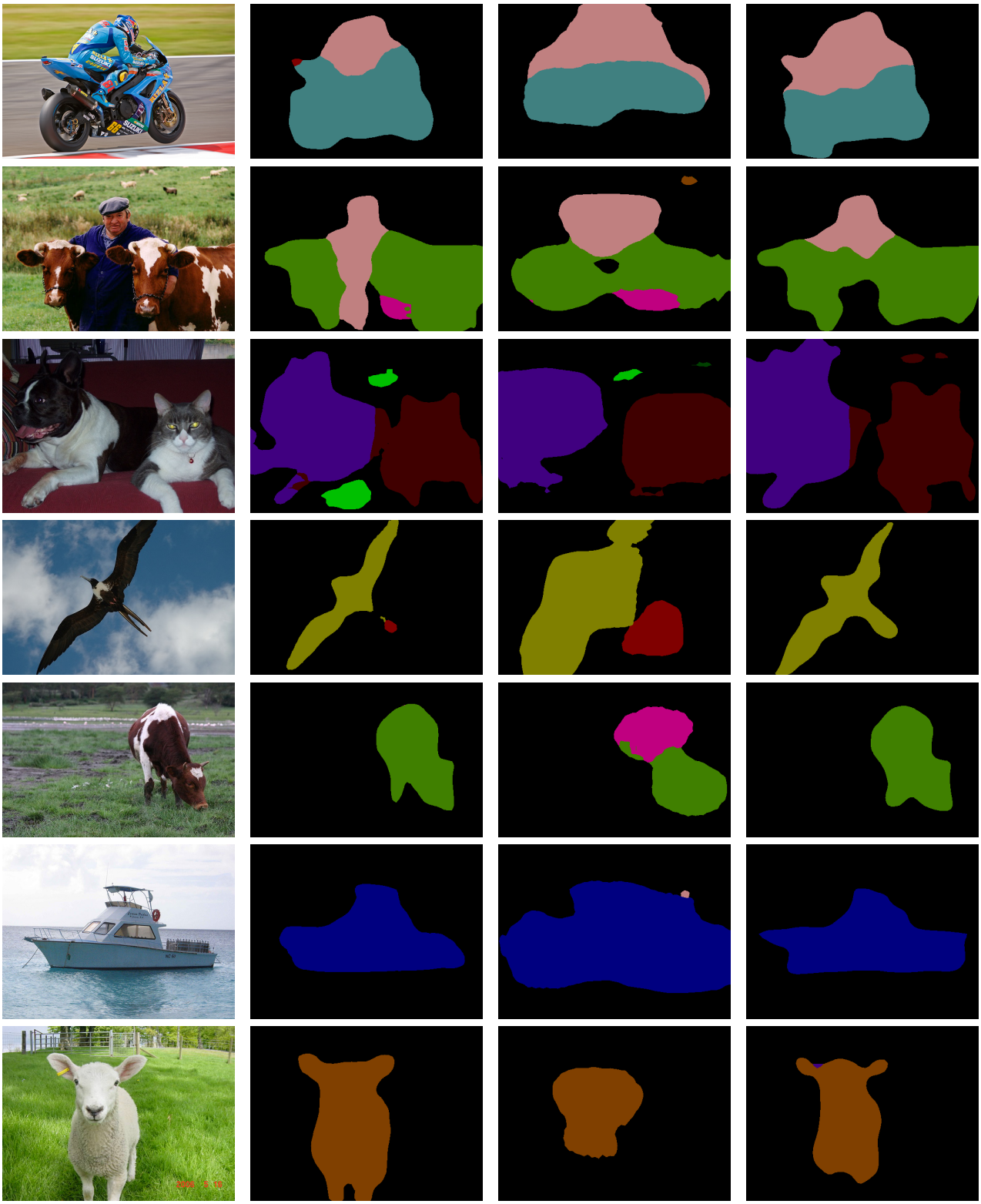


Image DeepLab [27] (full) EM-Adapt [126] (weak) M-CNN (weak)

Figure 4.4 – Comparison of fully-supervised DeepLab[27] (2nd column), weakly-supervised EM-Adapt [126] trained on VOC aug (3rd col.), and our weakly-supervised M-CNN trained on VOC+YouTube-Objects (4th col.), on VOC validation set. (*Best viewed in colour.*)

Method	Training data	# samples	Average	Average 10-class
Strong/Full supervision				
[134] + bb	VOC+ImNet	∼762,500	37.0	43.8
[134] + seg	VOC+ImNet	∼761,500	40.6	48.0
[126] + seg	VOC aug.	12,031	69.0	78.2
[112] (full)	VOC aug.	10,582	69.6	79.3
[202] (full)	VOC aug.+COCO	77,784	74.7	82.9
Weak supervision with additional info.				
[134] + sp	ImNet	∼760,000	35.8	42.3
[129] + sz	VOC aug.	10,582	43.3	48.9
[129] + sz + CRF	VOC aug.	10,582	45.1	51.2
[126] + CRF	VOC aug.	12,031	39.6	45.2
Weak supervision				
[130]	VOC aug.	12,031	25.7	-
[129]	VOC aug.	10,582	35.6	39.5
[126]	VOC aug.	12,031	35.2	40.3
Ours	VOC+YTube	3,139	**39.8**	**49.6**
Ours	VOC+ImNet	3,155	36.9	48.0

Table 4.1 – Performance of our M-CNN on the VOC 2012 test set is shown as IoU scores. We compare with several recent weakly-supervised methods: EM-Adapt [126], [130], [129], as well as methods using strong or full supervision: [134]+bb, [134]+seg, [126]+seg, [112], [202], and those using additional information: [134]+sp, [129]+sz, [129]+sz+CRF, [126]+CRF.

the impact of motion to learn segmentation. There is scope for further improvement, e.g., overcoming the confusion between similar classes in close proximity to each other, as in the challenging case in row 3 for cat vs dog.

Table 4.1 shows our evaluation on the VOC 2012 test set. We performed this by uploading our segmentation results to the evaluation server, as ground truth is not publicly available for the test set. We compare with several related methods as well. Note that all the scores reported for these methods in the table are taken directly from the publications, except [126] without the post-processing CRF step. This result, shown as '[126]' in the table, is with a model we trained on the VOC augmented dataset. We train M-CNN on all the 20 VOC classes with the model trained (and fine-tuned) on YouTube-Objects and perform a second fine-tuning step together with videos from YouTube-Objects and images from VOC. This achieves 39.8 mean IoU over all the 20 classes, and 49.6 on the 10 classes with video data. This result is significantly better than recent methods using only weak labels, which achieve 25.7 [130], 35.6 [129] and 35.2 [126]. The improvement shown by our M-CNN is more prominent when we compute the average over the 10 classes

where we use soft motion segmentation cues, with nearly 10% and 9% boost over [129] and [126] respectively. We also evaluated the model trained on ImageNet-VID instead of YouTube-Objects, which achieves 36.9 and 48.0 on 20 and 10 classes respectively.

A few methods have used additional information in the training process, such as the size of objects (+ sz in the table), superpixel segmentation (+ sp), or post-processing steps, e.g., introducing a CRF with pairwise terms learned from fully-annotated data (+ CRF), or even strong or full supervision, such as bounding box (+ bb) or pixel-level segmentation (+ seg) annotations. Even though our pure weakly-supervised method is not directly comparable to these approaches, we have included these results in the table for completeness. Nevertheless, M-CNN outperforms some of these methods [126], [134], due to our effective learning scheme. Also from Table 4.1, the number of training samples used for M-CNN (number of videos shots + number of VOC training images) is significantly lower than those for all the other methods.

4.1.3 Summary

We presented our M-CNN weakly-supervised learning approach for semantic segmentation, which uses only class labels assigned to videos. It integrates motion cues computed from video as soft constraints into a fully convolutional neural network. Experimental results show that our soft motion constraints can handle noisy motion information and improve significantly over the heuristic size constraints used by comparable approaches for weakly-supervised semantic segmentation, i.e., by EM-Adapt [126]. We show that this approach outperforms previous state of the art [126], [129] on the PASCAL VOC 2012 image segmentation dataset, thereby overcoming domain-shift issues typically seen when training on video and testing on images. More recent methods [71] have extended our M-CNN framework, closing the performance gap between fully and weakly-supervised segmentation approaches even further.

4.2 Incremental Learning

Modern detection methods, such as [14], [142], based on convolutional neural networks (CNNs) have achieved state-of-the-art results on benchmarks such as PASCAL VOC [43] and COCO [101]. This, however, comes with a high training time to learn the models. Furthermore, in an era where datasets are evolving regularly, with new classes and samples, it is necessary to develop incremental learning methods. A popular way to mitigate this is to use CNNs pretrained on a certain dataset for a task, and adapt them to new datasets or tasks, rather than train the entire network from scratch.

Fine-tuning [57] is one approach to adapt a network to new data or tasks. Here, the output layer of the original network is adjusted, either by replacing it with classes corresponding to the new task, or by adding new classes to the existing ones. The weights in this layer are then randomly initialized, and all the parameters of the network are tuned with the objective for the new task. While this framework is very successful on the new classes, its performance on the old ones suffers dramatically, if the network is not trained on all the classes jointly. This issue, where a neural network forgets previously learned knowledge when adapted to a new task, is referred to as catastrophic interference or forgetting. It has been known for over a couple of decades in the context of feedforward fully connected networks [109], [140], and needs to be addressed in the current state-of-the-art object detector networks, if we want to do incremental learning.

Consider the example in Figure 4.5. It illustrates catastrophic forgetting when incrementally adding a class, *horse* in this object detection example. The first CNN (top) is trained on three classes, including *person*, and localizes the rider in the image. The second CNN (bottom) is an incrementally trained version of the first one for the category *horse*. In other words, the original network is adapted with images from only this new class. This adapted network localizes the horse in the image, but fails to detect the rider, which it was capable of originally, and despite the fact that the *person* class was not updated. In this paper, we present a method to alleviate this issue.

Using only the training samples for the new classes, we propose a method for not only adapting the old network to the new classes, but also ensuring performance on the old classes does not degrade. The core of our approach is a loss function balancing the interplay between predictions on the new classes, i.e., cross-entropy loss, and a new distillation loss which minimizes the discrepancy between responses for old classes from the original and the new networks. The overall approach is illustrated in Figure 4.6.

We use a frozen copy of the original detection network to compute the distillation loss. This loss is related to the concept of "knowledge distillation" proposed in [69], but our application of it is significantly different from this previous work, as discussed in Section 4.2.1. We specifically target the problem of object detection, which has the additional challenge of localizing objects with bounding boxes, unlike other attempts [100], [141] limited to the image classification task. We demonstrate experimental results on the PASCAL VOC and COCO datasets using Fast R-CNN [56] as the network. Our results show that we can add new classes incrementally to an existing network without forgetting the original classes, and with no access to the original training data. We also evaluate variants of our method empirically, and show the influence of distillation and

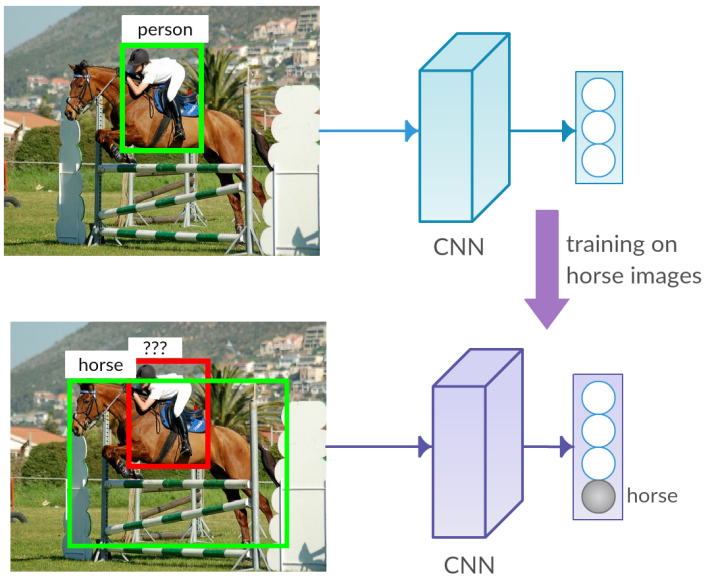


Figure 4.5 – Catastrophic forgetting. An object detector network originally trained for three classes, including *person*, detects the rider (top). When the network is retrained with images of the new class *horse*, it detects the horse in the test image, but fails to localize the rider (bottom).

the loss function. Note that our framework is general and can be applied to any other CNN-based object detectors where proposals are computed externally, or static sliding windows are used.

Related work. The problem of incremental learning has a long history in machine learning and artificial intelligence [26], [136], [155], [173]. Some of the more recent work, e.g., [29], [36], focuses on continuously updating the training set with data acquired from the Internet. They are: (i) restricted to learning with a fixed data representation [36], or (ii) keep all the collected data to retrain the model [29]. Other work partially addresses these issues by learning classifiers without access to the ensemble of data [110], [146], but uses a fixed image representation. Unlike these methods, our goal is learning the representation and classifiers jointly, without storing all the training examples.

Other related work in the context of deep learning, such as [100], uses knowledge distillation [69] for one of the classical vision tasks, image classification, formulated in a deep learning framework. However, their evaluation is limited to the case where the old network is trained on a dataset, while the new network is trained on a different one, e.g., Places365 for the old and PASCAL VOC for the new, ImageNet for the old and PASCAL VOC for the new, etc. Rebuffi et al. [141] also use knowledge distillation, but decouple the classifier and the representation learning. A more complete discussion on related work is available in [A25].

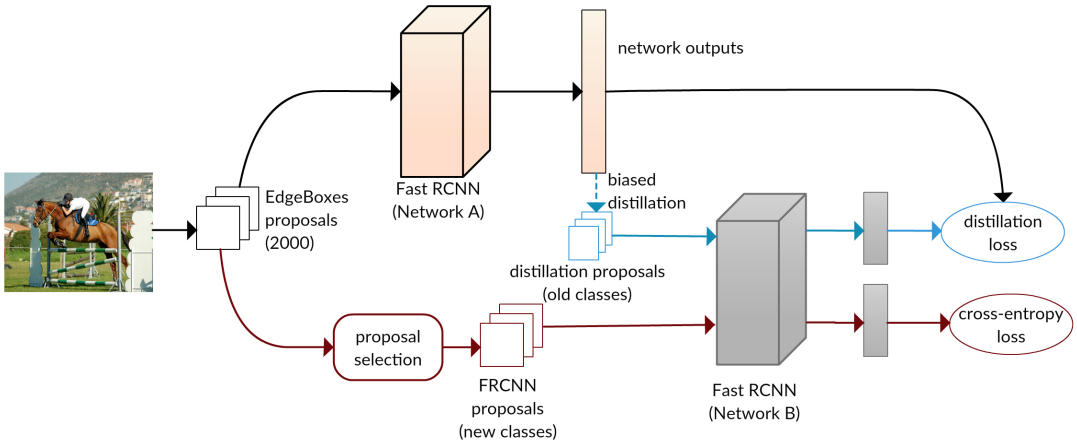


Figure 4.6 – Overview of our framework for learning object detectors incrementally. It is composed of a frozen copy of the detector (Network A) and the detector (Network B) adapted for the new class(es). See text for details.

4.2.1 Incremental learning of new classes

Our overall approach for incremental learning of a CNN model for object detection is illustrated in Figure 4.6. It contains a frozen copy of the original detector (denoted by Network A in the figure), which is used to: (i) select proposals corresponding to the old classes, i.e., distillation proposals, and (ii) compute the distillation loss. Network B in the figure is the adapted network for the new classes. It is obtained by increasing the number of outputs in the last layer of the original network, such that the new output layer includes the old as well as the new classes.

In order to avoid catastrophic forgetting, we constrain the learning process of the adapted network. We achieve this by incorporating a distillation loss, to preserve the performance on the old classes, as an additional term in the standard cross-entropy loss function (see Section 4.2.1). Specifically, we evaluate each new training sample on the frozen copy (Network A) to choose a diverse set of proposals (distillation proposals in Figure 4.6), and record their responses. With these responses in hand, we compute a distillation loss which measures the discrepancy between the two networks for the distillation proposals. This loss is added to the cross-entropy loss on the new classes to make up the loss function for training the adapted detection network. As we show in the experimental evaluation, the distillation loss as well as the strategy to select the distillation proposals are critical in preserving the performance on the old classes (see experimental results).

Object detection network. We use a variant of a popular framework for object detection—Fast R-CNN [56], which is a proposal-based detection method built with pre-computed object proposals, e.g., [10], [203]. We chose this instead of the more recent Faster R-CNN [142], which integrates the computation of category-specific proposals into

the network, because we need proposals agnostic to object categories, such as EdgeBoxes [203], MCG [10]. We use proposals from EdgeBoxes for PASCAL VOC 2007 and MCG for COCO. This allows us to focus on the problem of learning the representation and the classifier, given a pre-computed set of generic object proposals.

In our variant of Fast R-CNN, we replaced the VGG-16 trunk with a deeper ResNet-50 [67] component, which is faster and more accurate than VGG-16. We follow the suggestions in [67] to combine Fast R-CNN and ResNet architectures. The network processes the whole image through a sequence of residual blocks. Before the last strided convolution layer we insert a RoI pooling layer, which performs maxpooling over regions of varied sizes, i.e., proposals, into a 7×7 feature map. Then we add the remaining residual blocks, a layer for average pooling over spatial dimensions, and two fully connected layers: a softmax layer for classification (PASCAL or COCO classes, for example, along with the background class) and a regression layer for bounding box refinement, with independent corrections for each class.

The input to the network is an image and about 2000 precomputed object proposals represented as bounding boxes. During inference, the high-scoring proposals are refined according to bounding box regression. Then, a per-category non-maxima suppression (NMS) is performed to get the final detection results. The loss function to train the Fast R-CNN detector, corresponding to a RoI, is given by:

$$\mathcal{L}_{\text{rcnn}}(\mathbf{p}, k^*, t, t^*) = -\log \mathbf{p}_{k^*} + [k^* \geq 1] R(t - t^*), \tag{4.4}$$

where $\mathbf{p}$ is the set of responses of the network for all the classes (i.e., softmax output), k^* is a groundtruth class, t is an output of bounding box refinement layer, and $t*$ is the ground truth bounding box proposal. The first part of the loss denotes log-loss over classes, and the second part is localization loss. For more implementation details about Fast R-CNN, refer to the original paper [56].

Dual-network learning. First, we train a Fast R-CNN to detect the original set of classes C_A. We refer to this network as $\mathbf{A}(C_A)$. The goal now is to add a new set of classes C_B to this. We make two copies of $\mathbf{A}(C_A)$: one that is frozen to recognize classes C_A through distillation loss, and the second $\mathbf{B}(C_B)$ that is extended to detect the new classes C_B, which were not present or at least not annotated in the source images. The extension is done only in the last fully connected layers, i.e., classification and bounding box regression. We create sibling (i.e., fully-connected) layers [57] for new classes only and concatenate their outputs with the original ones. The new layers are initialized randomly in the same way as the corresponding layers in Fast R-CNN. Our goal is to train $\mathbf{B}(C_B)$ to recognize classes $C_A \cup C_B$ using only new data and annotations for C_B.

The distillation loss represents the idea of "keeping all the answers of the network the same or as close as possible". If we train $\mathbf{B}(C_B)$ without distillation, average precision on the old classes will degrade quickly, after a few hundred SGD iterations. This is a manifestation of catastrophic forgetting. We illustrate this in the results section. We compute the distillation loss by applying the frozen copy of $\mathbf{A}(C_A)$ to any new image. Even if no object is detected by $\mathbf{A}(C_A)$, the unnormalized logits (softmax input) carry

enough information to "distill" the knowledge of the old classes from $\mathbf{A}(C_A)$ to $\mathbf{B}(C_B)$. This process is illustrated in Figure 4.6.

For each image we randomly sample 64 RoIs out of 128 with the smallest background score. The logits computed for these RoIs by $\mathbf{A}(C_A)$ serve as targets for the old classes in the L_2 distillation loss shown below. The logits for the new classes C_B are not considered in this loss. We subtract the mean over the class dimension from these unnormalized logits (y) of each RoI to obtain the corresponding centered logits $\bar{y}$ used in the distillation loss. Bounding box regression outputs t_A (of the same set of proposals used for computing the logit loss) also constrain the loss of the network $\mathbf{B}(C_B)$. We chose to use L_2 loss instead of a cross-entropy loss for regression outputs because it demonstrates more stable training and performs better (see §4.2.2). The distillation loss combining the logits and regression outputs is written as:

$$\mathcal{L}_{\text{dist}}(y_A, t_A, y_B, t_B) = \frac{1}{N|C_A|} \sum \Big[(\bar{y}_A - \bar{y}_B)^2 + (t_A - t_B)^2 \Big], \tag{4.5}$$

where N is the number of RoIs sampled for distillation (i.e., 64 in this case), $|C_A|$ is the number of old classes, and the sum is over all the RoIs for the old classes. We distill logits without any smoothing, unlike [69], because most of the proposals already produce a smooth distribution of scores. Moreover, in our case, both the old and the new networks are similar with almost the same parameters (in the beginning), and so smoothing the logits distribution is not necessary to stabilize the learning.

The values of the bounding box regression are also distilled because we update all the layers, and any update of the convolutional layers will affect them indirectly. As box refinements are important to detect objects accurately, their values should be conserved as well. This is an easier task than keeping the classification scores because bounding box refinements for each class are independent, and are not linked by the softmax.

The overall loss $\mathcal{L}$ to train the model incrementally is a weighted sum of the distillation loss (4.5), and the standard Fast R-CNN loss (4.4) that is applied only to new classes C_B, where groundtruth bounding box annotation is available. In essence,

$$\mathcal{L} = \mathcal{L}_{\text{rcnn}} + \lambda \mathcal{L}_{\text{dist}}, \tag{4.6}$$

where the hyperparameter λ balances the two losses. We set λ to 1 in all the experiments with cross-validation.

The interplay between the two networks $\mathbf{A}(C_A)$ and $\mathbf{B}(C_B)$ provides the necessary supervision that prevents the catastrophic forgetting in the absence of original training data used by $\mathbf{A}(C_A)$. After the training of $\mathbf{B}(C_B)$ is completed, we can add more classes by freezing the newly trained network and using it for distillation. We can thus add new classes sequentially. Since $\mathbf{B}(C_B)$ is structurally identical to $\mathbf{A}(C_A \cup C_B)$, the extension can be repeated to add more classes.

Sampling strategy. As mentioned before, we choose 64 proposals out of 128 with the lowest background score, thus biasing the distillation to non-background proposals. We noticed that proposals recognized as confident background do not provide strong learning

method	old	new	all
A(1-10)	65.8	-	-
+**B**(11-20) w/o distillation	12.8	64.5	38.7
+**B**(11-20) w distillation	63.2	63.1	63.1
+**B**(11-20) w/o bbox distillation	58.7	63.1	60.9
+**B**(11-20) w EWC [85]	31.6	61.0	46.3
A(1-20)	68.4	71.3	69.8

Table 4.2 – **VOC 2007 test** average precision (%). Experiments demonstrating the addition of 10 classes, all at once, to a pretrained network. Classes 1-10 are the old classes, and 11-20 the new ones.

cues to conserve the original classes. One possibility is using an *unbiased distillation* that randomly samples 64 proposals out of the whole set of 2000 proposals. However, when doing so, the detection performance on old classes is noticeably worse because most of the distillation proposals are now background, and carry no strong signal about the object categories. Therefore, it is advantageous to select non-background proposals.

4.2.2 Results: Addition of multiple classes

We evaluate our method on the PASCAL VOC 2007 detection benchmark and the Microsoft COCO challenge dataset. This manuscript presents a selection of these results on VOC, and all the other analyses are presented in [A25]. We use the standard mean average precision (mAP) at 0.5 IoU threshold as the evaluation metric for our VOC experiments.

We train the network **A**(1-10) on the first 10 VOC classes (in alphabetical order) with the VOC trainval subset corresponding to these classes. In the second stage of training we used the remaining 10 classes as C_B and trained only on the images containing the new classes. Table 4.2 shows a summary of the evaluation of these networks on the VOC test set, with the full results in [A25].

Training the network **B**(11-20) on the 10 new classes with distillation (for the old classes) achieves 63.1% mAP ("**B**(11-20) w distillation" in the tables) compared to 69.8% of the baseline network trained on all the 20 classes ("**A**(1-20)"). Performance on the new classes is slightly worse than with the joint training of all the classes. The mAP on new classes is 63.1% for the network with distillation versus 71.3% for the jointly trained model. However, without distillation, the network achieves only 12.8% mAP ("+**B**(11-20) w/o distillation") on the old classes. Note that the method without bounding box distillation ("+**B**(11-20) w/o bbox distillation") is inferior to our full method ("+**B**(11-20) w distillation").

We also performed the 10-class experiment for different values of λ in (4.6), the hyperparameter controlling the relative importance of distillation and Fast R-CNN loss. Results shown in [A25] demonstrate that when the distillation is weak ($\lambda = 0.1$) the new classes are easier to learn, but the old ones are more easily forgotten. When distillation is strong ($\lambda = 10$), it destabilizes training and impedes learning the new classes. Set-

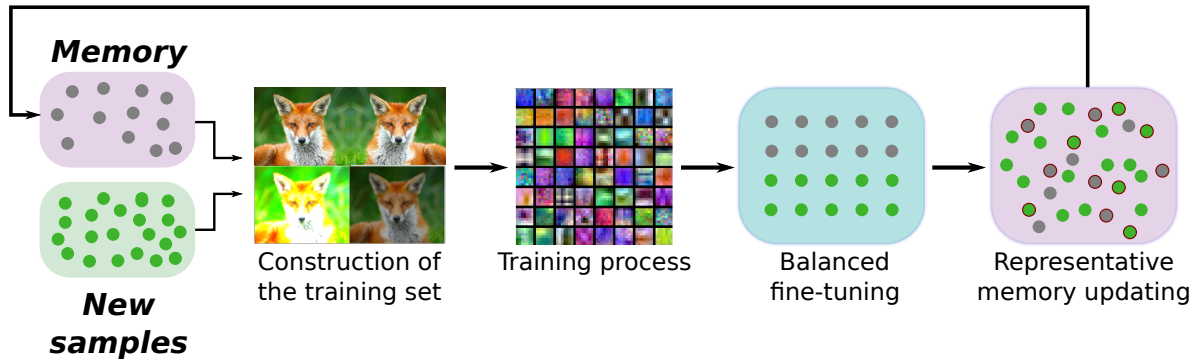


Figure 4.7 – Incremental training. Grey dots correspond to samples stored in the representative memory. Green dots correspond to samples from the new classes. Dots with red border correspond to the selected samples to be stored in the memory. (*Best viewed in colour.*)

ting λ to 1 is a good trade-off between learning new classes and preventing catastrophic forgetting.

We also compare our approach with elastic weight consolidation (EWC) [85], which is an alternative to distillation and applies per-parameter regularization selectively to alleviate catastrophic forgetting. We reimplemented EWC and verified that it produces results comparable to those reported in [85] on MNIST, and then adapted it to our object detection task. We do this by using the Fast R-CNN batches during the training phase, and by replacing log loss with the Fast R-CNN loss. Our approach outperforms EWC for this case, when we add 10 classes at once, as shown in Table 4.2.

Discussion. So far we have presented an approach for incremental learning of object detectors for new classes, without access to the training data corresponding to the old classes. We address the problem of catastrophic forgetting in this context, with a loss function that optimizes the performance on the new classes, in addition to preserving the performance on the old classes. Our recent work presented at ECCV 2018 [A9] studies a variant of this learning problem for image classification, when a few of the training samples corresponding to the old classes are available. An overview of this framework is illustrated in Figure 4.7 and explained in the following.

4.2.3 End-to-end incremental learning

Our end-to-end approach uses a deep network trained with a cross-distilled loss function, i.e., cross-entropy together with distillation loss. The network can be based on the architecture of most deep models designed for classification, since our approach does not require any specific properties. A typical architecture for classification can be seen in Fig. 4.8, with one *classification layer* and a classification loss. This *classification layer* uses features from the feature extractor to produce a set of *logits* which are transformed into class scores by a softmax layer (not shown in the figure). The only necessary modifi-

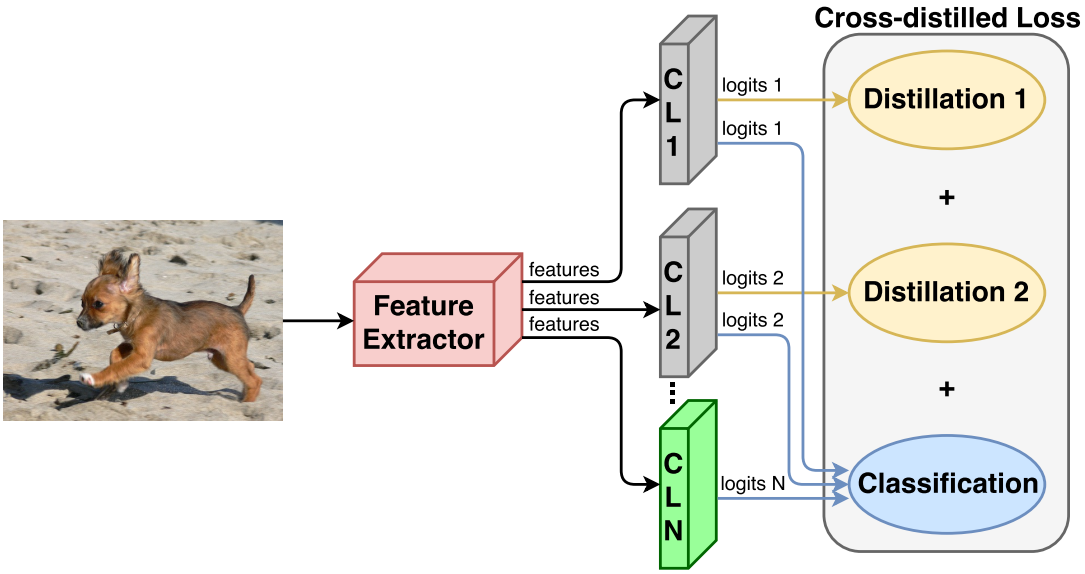


Figure 4.8 – Our incremental model. Given an input image, the feature extractor produces a set of features which are used by the *classification layers* (CLi blocks) to generate a set of *logits*. Grey *classification layers* contain old classes and their *logits* are used for distillation and classification. The green *classification layer* (CLN block) contains new classes and its *logits* are involved only in classification. (*Best viewed in colour.*)

cation is the loss function. To help our model retain the knowledge acquired from the old classes, we use a representative memory that stores and manages the most representative samples from the old classes. In addition to this we perform data augmentation and a balanced fine-tuning. All these components put together allow us to get state-of-the-art results.

Representative memory

When a new class or set of classes is added to the current model, a subset with the most representative samples from them is selected and stored in the representative memory. We investigate two memory setups in this work. The first setup considers a memory with a limited capacity of K samples. As the capacity of the memory is independent of the number of classes, the more classes stored, the fewer samples retained per class. The number of samples per class, n, is thus given by $n = \lfloor K/c \rfloor$, where c is the number of classes stored in memory, and K is the memory capacity. The second setup stores a constant number of exemplars per class. Thus, the size of the memory grows with the number of classes. The representative memory unit performs two operations: selection of new samples to store, and removal of leftover samples.

Selection of new samples. This is based on *herding selection* [188], which produces a sorted list of samples of one class based on the distance to the mean sample of that class. Given the sorted list of samples, the first n samples of the list are selected. These samples

are most representative of the class according to the mean. This selection method was chosen based on our experiments testing different approaches, such as random selection, histogram of the distances from each sample to the class mean. The selection is performed once per class, whenever a new class is added to the memory.

Removing samples. This step is performed after the training process to allocate memory for the samples from the new classes. As the samples are stored in a sorted list, this operation is trivial. The memory unit only needs to remove samples from the end of the sample set of each class. Note that after this operation, the removed samples are never used again.

Deep network

Architecture. The network is composed of several components, as illustrated in Fig. 4.8. The first component is a *feature extractor*, which is a set of layers to transform the input image into a feature vector. The next component is a *classification layer* which is the last fully-connected layer of the model, with as many outputs as the number of classes. This component takes the features and produces a set of *logits*. During the training phase, gradients to update the weights of the network are computed with these *logits* through our cross-distilled loss function. At test time, the loss function is replaced by a softmax layer (not shown in the figure).

To build our incremental learning framework, we start with a traditional, i.e., non-incremental, deep architecture for classification for the first set of classes. When new classes are trained, we add a new *classification layer* corresponding to these classes, and connect it to the *feature extractor* and the component for computing the cross-distilled loss, as shown in Fig. 4.8. Note that the architecture of the *feature extractor* does not change during the incremental training process, and only new *classification layers* are connected to it. Therefore, any architecture (or even pre-trained model) can be used with our approach just by adding the incremental classification layers and the cross-distilled loss function when necessary.

Cross-distilled loss function. This combines a distillation loss [69], which retains the knowledge from old classes, with a multi-class cross-entropy loss, which learns to classify the new classes. The distillation loss is applied to the *classification layers* of the old classes while the multi-class cross-entropy is used on all *classification layers*. This allows the model to update the decision boundaries of the classes. The loss computation is illustrated in Fig. 4.8. The cross-distilled loss function $L(\omega)$ is defined as:

$$L(\omega) = L_C(\omega) + \sum_{f=1}^{F} L_{D_f}(\omega), \tag{4.7}$$

where $L_C(\omega)$ is the cross-entropy loss applied to samples from the old and new classes, L_{D_f} is the distillation loss of the *classification layer* f, and F is the total number of *classification layers* for the old classes (shown as grey boxes in Fig. 4.8).

The cross-entropy loss $L_C(\omega)$ is given by:

$$L_C(\omega) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} p_{ij}\log q_{ij}, \tag{4.8}$$

where q_i is a score obtained by applying a softmax function to the *logits* of a *classification layer* for sample i, p_i is the ground truth for the sample i, and N and C denote the number of samples and classes respectively.

The distillation loss $L_D(\omega)$ is defined as:

$$L_D(\omega) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} pdist_{ij}\log qdist_{ij}, \tag{4.9}$$

where $pdist_i$ and $qdist_i$ are modified versions of p_i and q_i, respectively. They are obtained by raising p_i and q_i to the exponent $1/T$, as described in [69], where T is the distillation parameter. When $T = 1$, the class with the highest score influences the loss significantly, e.g., more than 0.9 from a maximum of 1.0, and the remaining classes with low scores have minimal impact on the loss. However, with $T > 1$, the remaining classes have a greater influence, and their higher loss values must be minimized. This forces the network to learn a more fine grained separation between them. As a result, the network learns a more discriminative representation of the classes. Based on our empirical results, we set T to 2 for all our experiments.

Results. We evaluate five different splits with different class order and incremental steps of 2, 5, 10, 20, and 50 classes. The class order is identical for all the evaluated methods, to ensure that the results are comparable. Table 4.3(a) summarizes the results of the experiments and Figure 4.9 shows the incremental steps for 2 and 5 classes. The rest of plots are included in the appendix **suppmat**. The 'Upper-Bound' result, shown in Figure 4.9 with a large cross (in magenta) in the last step, is obtained by training a non-incremental model using all the classes, and all their training samples.

We observe that our end-to-end approach obtains the best results for 2, 5, 10, and 20 classes. For 50 classes, although we achieve the same score as Hybrid1 (the variant of iCaRL using CNN classifier), we are 1% lower than iCaRL. This behaviour is due to the limited memory size, resulting in a heavily unbalanced training set containing 12.5 times more data from the new samples than from the old classes. To highlight the statistical significance of our method's performance compared to iCaRL, we performed a paired t-test on the results obtained for CIFAR-100. The corresponding p-values are 0.00005, 0.0005, 0.003, 0.0077, 0.9886 for 2, 5, 10, 20, and 50 classes respectively, which shows that the improvement of our method over iCaRL is statistically significant ($p < 0.01$) in all cases, except for 50 classes where both methods show similar performance.

It can be also observed that the performance of our approach remains stable across the incremental step sizes (from 2 to 20 classes per step) in Table 4.3(a), in contrast to all the other methods, which are dependent on the number of classes added in each step. This is because a small number of classes at each incremental step benefits the accuracy

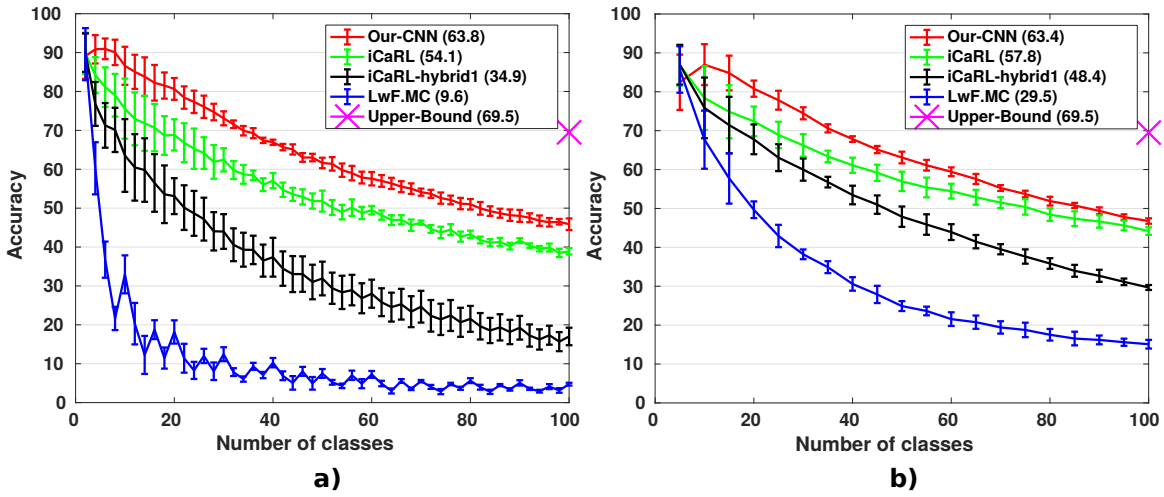


Figure 4.9 – **Accuracy on CIFAR-100.** Average and standard deviation of 5 executions with (a) 2 and (b) 5 classes per incremental step. Average of the incremental steps is shown in parentheses for each method. (Best viewed in pdf.)

# classes	2	5	10	20	50
Our-CNN	**63.8 ± 1.9**	**63.4 ± 1.6**	**63.6 ± 1.3**	**63.7 ± 1.1**	60.8 ± 0.3
iCaRL	54.1 ± 2.5	57.8 ± 2.6	60.5 ± 1.6	62.0 ± 1.2	**61.8 ± 0.4**
Hybrid1	34.9 ± 4.5	48.4 ± 2.6	55.8 ± 1.8	60.4 ± 1.0	60.8 ± 0.7
LwF.MC	9.6 ± 1.5	29.5 ± 2.2	40.4 ± 2.0	47.6 ± 1.5	52.9 ± 0.6

(a) CIFAR-100

# classes	10	100
Our-CNN	**90.4**	**69.4**
iCaRL	85.0	62.5
Hybrid1	83.5	46.1
LwF.MC	79.1	43.8

(b) ImageNet

Table 4.3 – Fixed memory size: accuracy on CIFAR-100 and ImageNet. Each column represents a different number of classes per incremental step. Each row represents a different approach. The best results are marked in bold.

in the early stages of the incremental learning process, as only a few classes must be classified. However, as more steps are applied to train all the classes, the accuracy of the final stages decreases.

The behaviour is reversed when larger number of classes are added in each incremental step. Lower accuracy values are seen during the early stages, but this is compensated with better values in the final stages. These effects can be seen in Figure 4.9, where two different number of classes per incremental step (2 and 5) are visualized. Figure 4.9 also shows that our approach is significantly better than iCaRL when a small number of classes per incremental step are employed. With larger number of classes in each

step, iCaRL approaches our performance, but still remains lower overall. Our approach clearly outperforms LwF.MC in all the cases, thus highlighting the importance of the representative memory in our model.

4.2.4 Summary

The two methods presented in this part of the chapter describe how CNNs can be trained in an incremental fashion using a combination of cross-entropy and distillation loss functions. In both cases, either no or very few samples from the old classes are used when the network is being updated with data from the new classes.

Chapter 5

Conclusion

This HDR manuscript has presented a selection of the author's collaborative research contributions during the past eight years. In particular, it focused on extracting priors as an additional cue for recognition problems in computer vision, where only partial annotations are available. Section 3.1 presented long-range motion cues for unsupervised segmentation. Tracking objects when only the first frame is annotated was discussed in Section 3.2, and Section 3.3 considered the case where an alternative form of supervision, through synthetic data, can be used for learning motion features. Chapter 4 presented two learning methods for weakly-supervised semantic segmentation (Section 4.1) and incremental learning, where none or a limited amount of the original data is available (Section 4.2). In addition to these partially-supervised methods, we discussed human and language-based priors for segmentation (Section 2.1) and pose estimation (Section 2.2), and text recognition (Section 2.3) respectively.

We will conclude the manuscript with a few directions for future work.

Modelling video data. Despite several successful models for image data, e.g., [93], their equivalents for video are still lacking. Popular methods are: (i) a simple extension of the bag-of-visual-words model to video, which is limited to capturing short-term motion patterns; (ii) ad-hoc video representations like dense trajectories [179] that model longer-term temporal relations in contrast to the bag-of-visual-words model, but require several hand-tuned parameters; and (iii) initial attempts for extending convolutional neural networks trained, at least in part, with images [164] to the video case, which are interesting, but do not fully exploit inherent motion cues. This problem of modelling video data is an ambitious goal, given the intrinsic variability of videos in both spatial and temporal dimensions. We plan to make progress in two directions to address this problem.

First, features will be learned directly from video data, and targeted specifically for each task defined on video, unlike previous work attempting to fit models trained on images to video data. Our approach will integrate additional cues, such as detection of people [64], [A13] and their body-joint locations [124], [125], and prior knowledge of the object of interest. The second direction builds on our work on automatically learning motion features from videos [A27], [A28]. In this related work [A27], [A28], we first

address the problem of determining whether an object is in motion, irrespective of the camera motion, and then learn its features. This methodology, however, was developed for video foreground/background segmentation, and now needs to be built for higher-level semantic tasks, such as action recognition, scene understanding. Recent work on actor-centric representations [169] is another avenue to explore, but should go beyond a simple box-based representation of actions.

Joint modelling of visual and audio data. With our new deep learning based video representation described above, the next objective is to find relations among visual and audio data samples. We plan to explore a graph matching framework [104], [194] in this context. Graph matching finds correspondences between vertices of two given graphs by optimizing a cost function. This framework has been popular for finding correspondences between a pair of images [A11], but now the objective is to go beyond such uni-modal matching to the challenging case of multi-modal matching, thereby leveraging the complementarity among different datasets.

In our case, an audio or a visual data sample denotes a graph and the components of data (e.g., audio or visual component features) represent its vertices. We plan to build on this basic framework with novel cost functions, incorporating recent advances in image and video segmentation [41], [55], and speech and text recognition [A16], [138]. Adapting the structure of the graph to each data sample, instead of imposing a single structure on the entire dataset is another research problem in this context. One possibility is to begin with a fully-connected graph, where the importance of each edge is represented by an additional learnt weight parameter. A weight close to zero implies that the edge connecting two nodes is not critical for matching. Recent work analyzing audio and image/video data jointly [8], [123], [201] is another interesting line of work to explore in this context. This joint analysis is achieved, for example, by learning a common embedding space where correspondences between audio and image data can be made directly [8], [123].

Incremental learning with synthetic data. We will explore the use of synthetic data within the incremental learning framework. Imbalance between the number of samples belonging to the old and the new classes is an important issue when learning a model incrementally [A9]. We propose to address this through the generation of synthetic samples. The synthetic examples will be generated using a selection of the old samples, in order to avoid storing the entire training set. Another important research question to be addressed here is the selection of old samples, including their number, and what criteria are relevant for the choice. A complementary direction in this research theme is the use of generative adversarial networks (GANs) [59] for data generation. GANs are popular for generating images today, but suffer from problems such as the lack of: (i) an explicit likelihood measure, which is commonplace in comparable generative models, and (ii) sufficient diversity in the generated data. Our recent work [A24] proposed ways to study these issues. The next step is to use our initial results to build stronger GAN models, with a particular focus on generating data for incremental learning.

Long-term memory modules. In the past few years there has been a growing interest in adding memory modules to machine learning algorithms, in particular, neural network models [61], [190], [193], building on classical ideas in [70], which implicitly encoded short-term memory by hidden states and weights. The memory component in recent neural network reads and writes observations from the past, and is used during both the training and test phases. All these previous attempts are, however, limited to text-based memory components, to answer questions about a sequence of sentences, e.g., a story [61], [190], [193], or simple decision oriented ones, for finding shortest paths or moves in a rule-based game [61]. None of them extend to learning image or video based memory components. One research direction to explore in this context is a visual memory network, with nodes to explicitly model memory. As part of this, we will introduce explicit memory elements, which store (and update) learnt representations of object categories and human actions of known classes. Separate units will store representations of unknown categories or actions, along with references to the ones they are most related to, for example.

Bibliography

[A1] K. Alahari and C. V. Jawahar, "Discriminative Actions for Recognising Events", in *ICVGIP*, 2006.

[A2] ——, "Dynamic Events as Mixtures of Spatial and Temporal Features", in *ICVGIP*, 2006.

[A3] K. Alahari, P. Kohli, and P. H. S. Torr, "Dynamic Hybrid Algorithms for MAP Inference in Discrete MRFs", *Trans. PAMI*, vol. 32, no. 10, pp. 1846–1857, 2010.

[A4] ——, "Reduce, Reuse & Recycle: Efficiently Solving Multi-Label MRFs", in *CVPR*, 2008.

[A5] K. Alahari, S. Kuthirummal, C. V. Jawahar, and P. J. Narayanan, "Geometric and Stochastic Error Minimisation in Motion Tracking", in *ACCV*, 2004.

[A6] K. Alahari, S. L. Putrevu, and C. V. Jawahar, "Discriminant Substrokes for Online Handwriting Recognition", in *ICDAR*, 2005.

[A7] ——, "Learning Mixtures of Offline and Online features for Handwritten Stroke Recognition", in *ICPR*, 2006.

[A8] K. Alahari, G. Seguin, J. Sivic, and I. Laptev, "Pose Estimation and Segmentation of People in 3D Movies", in *ICCV*, 2013.

[A9] F. M. Castro, M. J. Marin-Jimenez, N. Guil, C. Schmid, and K. Alahari, "End-to-End Incremental Learning", in *ECCV*, 2018.

[A10] A. Cherian, J. Mairal, K. Alahari, and C. Schmid, "Mixing Body-Part Sequences for Human Pose Estimation", in *CVPR*, 2014.

[A11] M. Cho, K. Alahari, and J. Ponce, "Learning Graphs to Match", in *ICCV*, 2013.

[A12] Y. Hua, K. Alahari, and C. Schmid, "Occlusion and Motion Reasoning for Long-term Tracking", in *ECCV*, 2014.

[A13] ——, "Online Object Tracking with Proposal Selection", in *ICCV*, 2015.

[A14] L. Ladicky, P. Sturgess, K. Alahari, C. Russell, and P. Torr, "What, Where & How Many? Combining Object Detectors and CRFs", in *ECCV*, 2010.

[A15] J. Lezama, K. Alahari, J. Sivic, and I. Laptev, "Track to the Future: Spatio-temporal Video Segmentation with Long-range Motion Cues", in *CVPR*, 2011.

[A16] A. Mishra, K. Alahari, and C. V. Jawahar, "Enhancing Energy Minimization Framework for Scene Text Recognition with Top-Down Cues", *CVIU*, vol. 145, pp. 30–42, 2016.

[A17] ——, "Image Retrieval using Textual Cues", in *ICCV*, 2013.

[A18] ——, "Scene Text Recognition using Higher Order Language Priors", in *BMVC*, 2012.

[A19] ——, "Top-Down and Bottom-Up Cues for Scene Text Recognition", in *CVPR*, 2012.

[A20] S. Ramalingam, P. Kohli, K. Alahari, and P. Torr, "Exact Inference in Multi-label CRFs with Higher Order Cliques", in *CVPR*, 2008.

[A21] U. Roy, A. Mishra, K. Alahari, and C. V. Jawahar, "Scene Text Recognition and Retrieval for Large Lexicons", in *ACCV*, 2014.

[A22] R. Sarvadevabhatla, K. Alahari, and C. V. Jawahar, "Recognizing Human Activities from Constituent Actions", in *National Conf. Communications*, 2005.

[A23] G. Seguin, K. Alahari, J. Sivic, and I. Laptev, "Pose Estimation and Segmentation of Multiple People in Stereoscopic Movies", *Trans. PAMI*, vol. 37, no. 8, pp. 1643–1655, 2015.

[A24] K. Shmelkov, C. Schmid, and K. Alahari, "How good is my GAN?", in *ECCV*, 2018.

[A25] ——, "Incremental Learning of Object Detectors without Catastrophic Forgetting", in *ICCV*, 2017.

[A26] P. Sturgess, K. Alahari, L. Ladicky, and P. H. S. Torr, "Combining Appearance and Structure from Motion Features for Road Scene Understanding", in *BMVC*, 2009.

[A27] P. Tokmakov, K. Alahari, and C. Schmid, "Learning Motion Patterns in Videos", in *CVPR*, 2017.

[A28] ——, "Learning Video Object Segmentation with Visual Memory", in *ICCV*, 2017.

[A29] ——, "Weakly-Supervised Semantic Segmentation using Motion Cues", in *ECCV*, 2016.

[A30] P. Tokmakov, C. Schmid, and K. Alahari, "Learning to Segment Moving Objects", *IJCV*, 2018 (In press).

[1] http://vision.middlebury.edu/stereo, 2013.

[2] http://pascallin.ecs.soton.ac.uk/challenges/{VOC}/voc2011.

[3] ICDAR 2003 datasets, http://algoval.essex.ac.uk/icdar.

[4] Street View Text dataset, http://vision.ucsd.edu/~kai/svt.

[5] E. H. Adelson, "On seeing stuff: the perception of materials by humans and machines", *Proc. SPIE*, vol. 4299, pp. 1–12, 2001.

[6] M. Andriluka, S. Roth, and B. Schiele, "People-tracking-by-detection and people-detection-by-tracking", in *CVPR*, 2008.

[7] ——, "Pictorial Structures Revisited: People Detection and Articulated Pose Estimation", in *CVPR*, 2009.

[8] R. Arandjelovic and A. Zisserman, "Objects that Sound", in *ECCV*, 2018.

[9] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, "Contour Detection and Hierarchical Image Segmentation", *Trans. PAMI*, vol. 33, no. 5, pp. 898–916, 2011.

[10] P. Arbeláez, J. Pont-Tuset, J. T. Barron, F. Marques, and J. Malik, "Multiscale Combinatorial Grouping", in *CVPR*, 2014.

[11] S. Avidan, "Ensemble Tracking", *Trans. PAMI*, vol. 29, no. 2, pp. 261–271, 2007.

[12] B. Babenko, M.-H. Yang, and S. Belongie, "Robust Object Tracking with Online Multiple Instance Learning", *Trans. PAMI*, vol. 33, no. 8, pp. 1619–1632, 2011.

[13] D. Batra, P. Yadollahpour, A. Guzman-Rivera, and G. Shakhnarovich, "Diverse M-best solutions in Markov random fields", in *ECCV*, 2012.

[14] S. Bell, C. L. Zitnick, K. Bala, and R. Girshick, "Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks", in *CVPR*, 2016.

[15] A. Bianne-Bernard, F. Menasri, R. A. Mohamad, C. Mokbel, C. Kermorvant, and L. Likforman-Sulem, "Dynamic and Contextual Information in HMM Modeling for Handwritten Word Recognition", *Trans. PAMI*, vol. 33, no. 10, pp. 2066–2080, 2011.

[16] P. Bideau and E. G. Learned-Miller, "It's Moving! A Probabilistic Model for Causal Motion Segmentation in Moving Camera Videos", in *ECCV*, 2016.

[17] A. Bissacco, M. Cummins, Y. Netzer, and H. Neven, "PhotoOCR: Reading Text in Uncontrolled Conditions", in *ICCV*, 2013.

[18] A. Blake, C. Rother, M. Brown, P. Perez, and P. H. S. Torr, "Interactive Image Segmentation Using an Adaptive GMMRF Model", in *ECCV*, 2004.

[19] E. Boros and P. L. Hammer, "Pseudo-Boolean optimization", *Discrete Applied Mathematics*, 2002.

[20] Y. Boykov and M.-P. Jolly, "Interactive Graph Cuts for Optimal Boundary and Region Segmentation of Objects in N-D Images", in *ICCV*, 2001.

[21] Y. Boykov, O. Veksler, and R. Zabih, "Fast Approximate Energy Minimization via Graph Cuts", *Trans. PAMI*, vol. 23, no. 11, pp. 1222–1239, 2001.

[22] W. Brendel and S. Todorovic, "Video object segmentation by tracking regions", in *ICCV*, 2009.

[23] M. Brox and J. Malik, "Object Segmentation by Long Term Analysis of Point Trajectories", in *ECCV*, 2010.

[24] T. Brox and J. Malik, "Large displacement optical flow: Descriptor matching in variational motion estimation", *Trans. PAMI*, vol. 33, no. 3, pp. 510–513, 2011.

[25] S. Caelles, K.-K. M. andJ. Pont-Tuset, L. Leal-Taixé, D. Cremers, and L. Van Gool, "One-Shot Video Segmentation", in *CVPR*, 2017.

[26] G. Cauwenberghs and T. Poggio, "Incremental and Decremental Support Vector Machine Learning", in *NIPS*, 2000.

[27] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected CRFs", in *ICLR*, 2015.

[28] ——, "DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs", *Trans. PAMI*, vol. 40, no. 4, pp. 834–848, 2018.

[29] X. Chen, A. Shrivastava, and A. Gupta, "NEIL: Extracting Visual Knowledge from Web Data", in *ICCV*, 2013.

[30] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis", *Trans. PAMI*, 2002.

[31] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection", in *CVPR*, 2005.

[32] M. Danelljan, G. Häger, F. S. Khan, and M. Felsberg, "Accurate Scale Estimation for Robust Visual Tracking", in *BMVC*, 2014.

[33] A. Delong, A. Osokin, H. N. Isack, and Y. Boykov, "Fast Approximate Energy Minimization with Label Costs", in *CVPR*, 2010.

[34] D. Dementhon, "Spatio-temporal segmentation of video by hierarchical mean shift analysis", in *Statistical Methods in Video Processing Workshop*, 2002.

[35] C. Desai, D. Ramanan, and C. Fowlkes, "Discriminative Models for Multi-class Object Layout", in *ICCV*, 2009.

[36] S. Divvala, A. Farhadi, and C. Guestrin, "Learning Everything about Anything: Webly-Supervised Visual Concept Learning", in *CVPR*, 2014.

[37] P. Dollár and C. L. Zitnick, "Structured Forests for Fast Edge Detection", in *ICCV*, 2013.

[38] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazırbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox, "FlowNet: Learning Optical Flow with Convolutional Networks", in *ICCV*, 2015.

[39] M. Eichner, M. Marin-Jimenez, A. Zisserman, and V. Ferrari, "2D Articulated Human Pose Estimation and Retrieval in (Almost) Unconstrained Still Images", *IJCV*, vol. 99, no. 2, pp. 190–214, 2012.

[40] K. Elagouni, C. Garcia, and P. Sébillot, "A comprehensive neural-based approach for text recognition in videos using natural language processing", in *ICMR*, 2011.

[41] I. Endres and D. Hoiem, "Category-Independent Object Proposals with Diverse Ranking", *Trans. PAMI*, vol. 36, no. 2, pp. 222–234, 2014.

[42] B. Epshtein, E. Ofek, and Y. Wexler, "Detecting Text in Natural Scenes with Stroke Width Transform", in *CVPR*, 2010.

[43] M. Everingham, S. M. A. Eslami, L. van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes Challenge: A Retrospective", *IJCV*, vol. 111, no. 1, pp. 98–136, 2015.

[44] M. Everingham, J. Sivic, and A. Zisserman, "'Hello! My name is... Buffy' - Automatic naming of characters in TV video", in *BMVC*, 2006.

[45] A. Faktor and M. Irani, "Video Segmentation by Non-Local Consensus Voting", in *BMVC*, 2014.

[46] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling", *Trans. PAMI*, vol. 35, no. 8, pp. 1915–1929, 2013.

[47] J. L. Feild and E. G. Learned-Miller, "Improving Open-Vocabulary Scene Text Recognition", in *ICDAR*, 2013.

[48] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient Graph-Based Image Segmentation", *IJCV*, vol. 59, no. 2, pp. 167–181, 2004.

[49] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan, "Object Detection with Discriminatively Trained Part Based Models", *Trans. PAMI*, vol. 32, no. 9, pp. 1627–1645, 2010.

[50] P. Felzenszwalb and D. Huttenlocher, "Pictorial structures for object recognition", *IJCV*, vol. 61, no. 1, pp. 55–79, 2005.

[51] P. Felzenszwalb and D. Huttenlocher, "Distance Transforms of Sampled Functions", *Theory of Computing*, vol. 8, no. 19, 2012.

[52] V. Ferrari, M. Marin-Jimenez, and A. Zisserman, "Progressive search space reduction for human pose estimation", in *CVPR*, 2008.

[53] M. Fischler and R. Elschlager, "The representation and matching of pictorial structures", *IEEE Trans. Computers*, vol. 100, no. 1, pp. 67–92, 1973.

[54] K. Fragkiadaki, H. Hu, and J. Shi, "Pose from Flow and Flow from Pose", in *CVPR*, 2013.

[55] J. Gao, Z. Yang, C. Sun, K. Chen, and R. Nevatia, "TURN TAP: Temporal Unit Regression Networks for Temporal Action Proposals", in *ICCV*, 2017.

[56] R. Girshick, "Fast R-CNN", in *ICCV*, 2015.

[57] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation", in *CVPR*, 2014.

[58] L. Gómez and D. Karatzas, "Scene Text Recognition: No Country for Old Men?", in *ACCV Workshops*, 2014.

[59] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets", in *NIPS*, 2014.

[60] A. Graves, N. Jaitly, and A. Mohamed, "Hybrid speech recognition with deep bidirectional LSTM", in *Workshop on Automatic Speech Recognition and Understanding*, 2013.

[61] A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwińska, S. G. Colmenarejo, E. Grefenstette, T. Ramalho, J. Agapiou, A. P. Badia, K. M. Hermann, Y. Zwols, G. Ostrovski, A. Cain, H. King, C. Summerfield, P. Blunsom, K. Kavukcuoglu, and D. Hassabis, "Hybrid computing using a neural network with dynamic external memory", *Nature*, 2016.

[62] M. Grundmann, V. Kwatra, M. Han, and I. Essa, "Efficient Hierarchical Graph-Based Video Segmentation", in *CVPR*, 2010.

[63] V. Gulshan, V. Lempitsky, and A. Zisserman, "Humanising GrabCut: Learning to segment humans using the Kinect", in *ICCV Workshop Consumer Depth Cameras for Computer Vision*, 2011.

[64] B. Han, H. Adam, and J. Sim, "BranchOut: Regularization for Online Ensemble Tracking with CNNs", in *CVPR*, 2017.

[65] S. Hare, A. Saffari, and P. H. S. Torr, "Struck: Structured output tracking with kernels", in *ICCV*, 2011.

[66] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004.

[67] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition", in *CVPR*, 2016.

[68] J. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-Speed Tracking with Kernelized Correlation Filters", *Trans. PAMI*, vol. 37, no. 3, pp. 583–596, 2015.

[69] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network", in *NIPS*, 2014.

[70] S. Hochreiter and J. Schmidhuber, "Long Short-term Memory", *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[71] S. Hong, D. Yeo, S. Kwak, H. Lee, and B. Han, "Weakly Supervised Semantic Segmentation using Web-Crawled Videos", in *CVPR*, 2017.

[72] P. V. C. Hough, *Method and means for recognizing complex patterns*, US Patent 3,069,654, 1962.

[73] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "Flownet 2.0: Evolution of optical flow estimation with deep networks", in *CVPR*, 2017.

[74] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman, "Reading Text in the Wild with Convolutional Neural Networks", *IJCV*, vol. 116, no. 1, pp. 1–20, 2016.

[75] M. Jaderberg, A. Vedaldi, and A. Zisserman, "Deep Features for Text Spotting", in *ECCV*, 2014.

[76] S. Jain, B. Xiong, and K. Grauman, "Fusionseg: Learning to combine motion and appearance for fully automatic segmention of generic objects in videos", in *CVPR*, 2017.

[77] T. Judd, K. Ehinger, F. Durand, and A. Torralba, "Learning to Predict Where Humans Look", in *ICCV*, 2009.

[78] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-Learning-Detection", *Trans. PAMI*, vol. 34, no. 7, pp. 1409–1422, 2012.

[79] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems", *J. Fluids Engineering*, 1960.

[80] D. Karatzas, F. Shafait, S. Uchida, M. Iwamura, L. G. i Bigorda, S. R. Mestre, J. Mas, D. F. Mota, J. Almazán, and L. de las Heras, "ICDAR 2013 Robust Reading Competition", in *ICDAR*, 2013.

[81] C. Keller, M. Enzweiler, M. Rohrbach, D. Llorca, C. Schnorr, and D. Gavrila, "The Benefits of Dense Stereo for Pedestrian Detection", *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 4, pp. 1096–1106, 2011.

[82] M. Keuper, B. Andres, and T. Brox, "Motion trajectory segmentation via minimum cost multicuts", in *ICCV*, 2015.

[83] A. Khoreva, F. Galasso, M. Hein, and B. Schiele, "Classifier Based Graph Construction for Video Segmentation", in *CVPR*, 2015.

[84] A. Khoreva, F. Perazzi, R. Benenson, B. Schiele, and A. Sorkine-Hornung, "Learning Video Object Segmentation from Static Images", in *CVPR*, 2017.

[85] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell, "Overcoming catastrophic forgetting in neural networks", *PNAS*, 2017.

[86] Y. J. Koh and C.-S. Kim, "Primary Object Segmentation in Videos Based on Region Augmentation and Reduction", in *CVPR*, 2017.

[87] V. Kolmogorov, A. Criminisi, A. Blake, G. Cross, and C. Rother, "Bi-layer segmentation of binocular stereo video", in *CVPR*, 2005.

[88] V. Kolmogorov and R. Zabih, "What energy functions can be minimized via graph cuts", *Trans. PAMI*, vol. 26, no. 2, pp. 147–159, 2004.

[89] V. Kolmogorov, "Convergent Tree-Reweighted Message Passing for Energy Minimization", *Trans. PAMI*, vol. 28, no. 10, pp. 1568–1583, 2006.

[90] N. Komodakis, N. Paragios, and G. Tziritas, "MRF Optimization via Dual Decomposition: Message-Passing Revisited", in *ICCV*, 2007.

[91] S. Koppal, C. Zitnick, M. Cohen, S. Kang, B. Ressler, and A. Colburn, "A viewer-centric editor for 3D movies", *Computer Graphics and Applications*, vol. 31, no. 1, pp. 20–35, 2011.

[92] P. Krähenbühl and V. Koltun, "Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials", in *NIPS*, 2011.

[93] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks", in *NIPS*, 2012.

[94] D. Kumar, M. N. A. Prasad, and A. G. Ramakrishnan, "NESP: Nonlinear enhancement and selection of plane for optimal segmentation and recognition of scene word images", in *DRR*, 2013.

[95] M. P. Kumar, P. H. S. Torr, and A. Zisserman, "Learning Layered Motion Segmentations of Video", in *ICCV*, 2005.

[96] L. Ladicky, P. H. S. Torr, and A. Zisserman, "Human Pose Estimation using a Joint Pixel-wise and Part-wise Formulation", in *CVPR*, 2013.

[97] J. Lafferty, A. McCallum, and F. Pereira, "Conditional Random Fields: Probabilistic models for segmenting and labelling sequence data", in *ICML*, 2001.

[98] M. W. Lee and R. Nevatia, "Human pose tracking in monocular sequence using multilevel structured models", *Trans. PAMI*, vol. 31, no. 1, pp. 27–38, 2009.

[99] Y. Lee, J. Kim, and K. Grauman, "Key-segments for video object segmentation", in *ICCV*, 2011.

[100] Z. Li and D. Hoiem, "Learning without forgetting", in *ECCV*, 2016.

[101] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context", in *ECCV*, 2014.

[102] C. Liu, "Beyond Pixels: Exploring New Representations and Applications for Motion Analysis", PhD thesis, Massachusetts Institute of Technology, 2009.

[103] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation", in *CVPR*, 2015.

[104] L. Lovász and M. D. Plummer, *Matching theory*. Amsterdam, New York: North-Holland, 1986.

[105] M. Kristan et al., "The Visual Object Tracking VOT2014 challenge results", in *ECCV Visual Object Tracking Challenge Workshop*, 2014.

[106] M. Marszałek, I. Laptev, and C. Schmid, "Actions in Context", in *CVPR*, 2009.

[107] I. Matthews, T. Ishikawa, and S. Baker, "The Template Update Problem", *Trans. PAMI*, vol. 26, no. 6, pp. 810–815, 2004.

[108] N. Mayer, E. Ilg, P. Häusser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox, "A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation", in *CVPR*, 2016.

[109] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem", *Psychology of learning and motivation*, vol. 24, pp. 109–165, 1989.

[110] T. Mensink, J. Verbeek, F. Perronnin, and G. Csurka, "Distance-Based Image Classification: Generalizing to New Classes at Near-Zero Cost", *Trans. PAMI*, vol. 35, no. 11, pp. 2624–2637, 2013.

[111] A. Monroy and B. Ommer, "Beyond Bounding-Boxes: Learning Object Shape by Model-Driven Grouping", in *ECCV*, 2012.

[112] M. Mostajabi, P. Yadollahpour, and G. Shakhnarovich, "Feedforward Semantic Segmentation With Zoom-Out Features", in *CVPR*, 2015.

[113] K. P. Murphy, Y. Weiss, and M. I. Jordan, "Loopy belief propagation for approximate inference: An empirical study", in *UAI*, 1999.

[114] G. Nagy, "Twenty Years of Document Image Analysis in PAMI", *Trans. PAMI*, vol. 22, no. 1, pp. 38–62, 2000.

[115] M. Narayana, A. R. Hanson, and E. G. Learned-Miller, "Coherent Motion Segmentation in Moving Camera Videos Using Optical Flow Orientations", in *ICCV*, 2013.

[116] G. Nebehay and R. Pflugfelder, "Consensus-based matching and tracking of keypoints for object tracking", in *WACV*, 2014.

[117] L. Neumann and J. Matas, "A Method for Text Localization and Recognition in Real-World Images", in *ACCV*, 2010.

[118] ——, "A real-time scene text to speech system", in *ECCV workshops*, 2012.

[119] ——, "On Combining Multiple Segmentations in Scene Text Recognition", in *ICDAR*, 2013.

[120] ——, "Real-time scene text localization and recognition", in *CVPR*, 2012.

[121] J. C. Niebles, B. Han, and L. Fei-Fei, "Efficient Extraction of Human Motion Volumes by Tracking", in *CVPR*, 2010.

[122] T. Novikova, O. Barinova, P. Kohli, and V. S. Lempitsky, "Large-Lexicon Attribute-Consistent Text Recognition in Natural Images", in *ECCV*, 2012.

[123] A. Owens and A. A. Efros, "Audio-Visual Scene Analysis with Self-Supervised Multisensory Features", in *ECCV*, 2018.

[124] G. Papandreou, T. Zhu, L. Chen, S. Gidaris, J. Tompson, and K. Murphy, "PersonLab: Person Pose Estimation and Instance Segmentation with a Part-Based Geometric Embedding Model", in *ECCV*, 2018.

[125] G. Papandreou, T. Zhu, N. Kanazawa, A. Toshev, J. Tompson, C. Bregler, and K. Murphy, "Towards Accurate Multi-person Pose Estimation in the Wild", in *CVPR*, 2017.

[126] G. Papandreou, L.-C. Chen, K. Murphy, and A. L. Yuille, "Weakly-and semi-supervised learning of a DCNN for semantic image segmentation", in *ICCV*, 2015.

[127] A. Papazoglou and V. Ferrari, "Fast object segmentation in unconstrained video", in *ICCV*, 2013.

[128] D. Park and D. Ramanan, "N-best maximal decoders for part models", in *ICCV*, 2011.

[129] D. Pathak, P. Krähenbühl, and T. Darrell, "Constrained Convolutional Neural Networks for Weakly Supervised Segmentation", in *ICCV*, 2015.

[130] D. Pathak, E. Shelhamer, J. Long, and T. Darrell, "Fully convolutional multi-class multiple instance learning", in *ICLR*, 2015.

[131] J. Pearl, *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference*. Morgan Kauffman, 1988.

[132] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. van Gool, M. Gross, and A. Sorkine-Hornung, "A Benchmark Dataset and Evaluation Methodology for Video Object Segmentation", in *CVPR*, 2016.

[133] P. Pinheiro, T.-Y. Lin, R. Collobert, and P. Dollár, "Learning to refine object segments", in *ECCV*, 2016.

[134] P. O. Pinheiro and R. Collobert, "From Image-level to Pixel-level Labeling with Convolutional Networks", in *CVPR*, 2015.

[135] J. C. Platt, "Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods", in *Advances in Large Margin Classifiers*, 1999.

[136] R. Polikar, L. Upda, S. S. Upda, and V. Honavar, "Learn++: An incremental learning algorithm for supervised neural networks", *IEEE Trans. Systems, Man, and Cybernetics, Part C*, vol. 31, no. 4, pp. 497–508, 2001.

[137] A. Prest, C. Leistner, J. Civera, C. Schmid, and V. Ferrari, "Learning object class detectors from weakly annotated video", in *CVPR*, 2012.

[138] G. Pundak, T. Sainath, R. Prabhavalkar, A. Kannan, and D. Zhao, "Deep context: End-to-end contextual speech recognition", in *IEEE Spoken Lang. Tech.*, 2018.

[139] D. Ramanan, D. A. Forsyth, and A. Zisserman, "Strike a pose: Tracking people by finding stylized poses", in *CVPR*, 2005.

[140] R. Ratcliff, "Connectionist models of recognition memory: constraints imposed by learning and forgetting functions.", *Psychological review*, vol. 97, no. 2, p. 285, 1990.

[141] S.-A. Rebuffi, A. Kolesnikov, and C. H. Lampert, "iCaRL: Incremental Classifier and Representation Learning", in *CVPR*, 2017.

[142] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", in *NIPS*, 2015.

[143] X. Ren, L. Bo, and D. Fox, "RGB-(D) Scene Labeling: Features and Algorithms", in *CVPR*, 2012.

[144] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid, "EpicFlow: Edge-Preserving Interpolation of Correspondences for Optical Flow", in *CVPR*, 2015.

[145] E. M. Riseman and A. R. Hanson, "A Contextual Postprocessing System for Error Correction Using Binary n-Grams", *IEEE Trans. Comput.*, pp. 480–493, 1974.

[146] M. Ristin, M. Guillaumin, J. Gall, and L. V. Gool, "Incremental Learning of NCM Forests for Large-Scale Image Classification", in *CVPR*, 2014.

[147] M. Rohrbach, S. Amin, M. Andriluka, and B. Schiele, "A database for fine grained activity detection of cooking activities", in *CVPR*, 2012.

[148] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation", in *MICCAI*, 2015.

[149] C. Rother, V. Kolmogorov, and A. Blake, "Grabcut: Interactive foreground extraction using iterated graph cuts", *ACM Trans. Graphics*, vol. 23, no. 3, pp. 309–314, 2004.

[150] B. Russell, A. Efros, J. Sivic, W. T. Freeman, and A. Zisserman, "Using Multiple Segmentations to Discover Objects and their Extent in Image Collections", in *CVPR*, 2006.

[151] C. Russell, L. Ladicky, P. Kohli, and P. H. S. Torr, "Exact and Approximate Inference in Associative Hierarchical Networks using Graph Cuts", in *UAI*, 2010.

[152] P. Sand and S. Teller, "Particle Video: Long-Range Motion Estimation Using Point Trajectories", *IJCV*, vol. 80, no. 1, 2008.

[153] B. Sapp, A. Toshev, and B. Taskar, "Cascaded models for articulated pose estimation", in *ECCV*, 2010.

[154] B. Sapp, D. Weiss, and B. Taskar, "Parsing human motion with stretchable models", in *CVPR*, 2011.

[155] J. C. Schlimmer and D. H. Fisher, "A Case Study of Incremental Concept Induction", in *AAAI*, 1986.

[156] A. Shahab, F. Shafait, and A. Dengel, "ICDAR 2011 Robust Reading Competition Challenge 2: Reading Text in Scene Images", in *ICDAR*, 2011.

[157] G. Sheasby, J. Valentin, N. Crook, and P. H. S. Torr, "A Robust Stereo Prior for Human Segmentation", in *ACCV*, 2012.

[158] C. Shi, C. Wang, B. Xiao, Y. Zhang, S. Gao, and Z. Zhang, "Scene Text Recognition Using Part-Based Tree-Structured Character Detection", in *CVPR*, 2013.

[159] J. Shi and J. Malik, "Motion segmentation and tracking using normalized cuts", in *ICCV*, 1998.

[160] ——, "Normalized Cuts and Image Segmentation", *Trans. PAMI*, vol. 22, no. 8, pp. 888–905, 2000.

[161] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, "Real-time human pose recognition in parts from single depth images", in *CVPR*, 2011.

[162] J. Shotton, J. Winn, C. Rother, and A. Criminisi, "TextonBoost for Image Understanding: Multi-Class Object Recognition and Segmentation by Jointly Modeling Texture, Layout, and Context", *IJCV*, vol. 81, no. 1, pp. 2–23, 2009.

[163] H. Sidenbladh, M. Black, and D. Fleet, "Stochastic Tracking of 3D Human Figures Using 2D Image Motion", in *ECCV*, 2000.

[164] K. Simonyan and A. Zisserman, "Two-Stream Convolutional Networks for Action Recognition in Videos", in *NIPS*, 2014.

[165] ——, "Very Deep Convolutional Networks for Large-Scale Image Recognition", in *ICLR*, 2015.

[166] C. Sminchisescu and B. Triggs, "Estimating articulated human motion with covariance scaled sampling", *IJRR*, vol. 22, no. 6, pp. 371–391, 2003.

[167] C. Sminchisescu and A. Jepson, "Variational mixture smoothing for non-linear dynamical systems", in *CVPR*, 2004.

[168] A. Stein, D. Hoiem, and M. Hebert, "Learning to Extract Object Boundaries using Motion Cues", in *ICCV*, 2007.

[169] C. Sun, A. Shrivastava, C. Vondrick, K. Murphy, R. Sukthankar, and C. Schmid, "Actor-centric Relation Network", in *ECCV*, 2018.

[170] N. Sundaram, T. Brox, and K. Keutzer, "Dense point trajectories by GPU-accelerated large displacement optical flow", in *ECCV*, 2010.

[171] J. S. Supancic and D. Ramanan, "Self-Paced Learning for Long-Term Tracking", in *CVPR*, 2013.

[172] C. Thillou, S. Ferreira, and B. Gosselin, "An embedded application for degraded text recognition", *EURASIP J. Applied Signal Processing*, pp. 2127–2135, 2005.

[173] S. Thrun, "Is learning the n-th thing any easier than learning the first?", in *NIPS*, 1996.

[174] C. Tomasi and T. Kanade, "Detection and tracking of point features", Carnegie Mellon University School of Computer Science, Tech. Rep. CMU-CS-91132, 1991.

[175] A. Vazquez-Reina, S. Avidan, H. Pfister, and E. Miller, "Multiple Hypothesis Video Segmentation from Superpixel Flows", in *ECCV*, 2010.

[176] A. Vedaldi and A. Zisserman, "Efficient Additive Kernels via Explicit Feature Maps", *Trans. PAMI*, vol. 34, no. 3, pp. 480–492, 2012.

[177] A. Vezhnevets, V. Ferrari, and J. M. Buhmann, "Weakly supervised structured output learning for semantic segmentation", in *CVPR*, 2012.

[178] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features", in *CVPR*, 2001.

[179] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu, "Dense trajectories and motion boundary descriptors for action recognition", *IJCV*, vol. 103, no. 1, pp. 60–79, 2013.

[180] J. M. Wang, D. J. Fleet, and A. Hertzmann, "Gaussian process dynamical models for human motion", *Trans. PAMI*, 2008.

[181] K. Wang, B. Babenko, and S. Belongie, "End-to-End Scene Text recognition", in *ICCV*, 2011.

[182] J. Weinman, Z. Butler, D. Knoll, and J. Feild, "Toward Integrated Scene Text Reading", *Trans. PAMI*, vol. 36, no. 2, pp. 375–387, 2014.

[183] J. J. Weinman, E. G. Learned-Miller, and A. R. Hanson, "Scene Text Recognition Using Similarity and a Lexicon with Sparse Belief Propagation", *Trans. PAMI*, vol. 31, no. 10, pp. 1733–1746, 2009.

[184] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid, "Learning to Detect Motion Boundaries", in *CVPR*, 2015.

[185] D. Weiss, B. Sapp, and B. Taskar, "Sidestepping intractable inference with structured ensemble cascades", in *NIPS*, 2010.

[186] Y. Weiss, "Smoothness in layers: Motion segmentation using nonparametric mixture estimation", in *CVPR*, 1997.

[187] Y. Weiss, "Correctness of local probability propagation in graphical models with loops", *Neural computation*, 2000.

[188] M. Welling, "Herding Dynamical Weights to Learn", in *ICML*, 2009.

[189] P. Werbos, "Backpropagation through time: What it does and how to do it", *Proc. IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[190] J. Weston, S. Chopra, and A. Bordes, "Memory Networks", in *ICLR*, 2015.

[191] J. Wu, Y. Zhao, J. Zhu, S. Luo, and Z. Tu, "MILCut: A Sweeping Line Multiple Instance Learning Paradigm for Interactive Image Segmentation", in *CVPR*, 2014.

[192] Y. Wu, J. Lim, and M.-H. Yang, "Online Object Tracking: A Benchmark", in *CVPR*, 2013.

[193] C. Xiong, S. Merity, and R. Socher, "Dynamic Memory Networks for Visual and Textual Question Answering", in *ICML*, 2016.

[194] J. Yan, M. Cho, H. Zha, X. Yang, and S. M. Chu, "Multi-Graph Matching via Affinity Optimization with Graduated Consistency Regularization", *Trans. PAMI*, 2016.

[195] Y. Yang, S. Hallman, D. Ramanan, and C. Fowlkes, "Layered Object Models for Image Segmentation", *Trans. PAMI*, vol. 34, no. 9, pp. 1731–1743, 2011.

[196] Y. Yang and D. Ramanan, "Articulated Human Detection with Flexible Mixtures-of-Parts", *Trans. PAMI*, 2012.

[197] ——, "Articulated Pose Estimation using Flexible Mixtures of Parts", in *CVPR*, 2011.

[198] B. Yao and L. Fei-Fei, "Modeling Mutual Context of Object and Human Pose in Human-Object Interaction Activities", in *CVPR*, 2010.

[199] C. Yao, X. Bai, B. Shi, and W. Liu, "Strokelets: A Learned Multi-scale Representation for Scene Text Recognition", in *CVPR*, 2014.

[200] Q. Ye and D. Doermann, "Text Detection and Recognition in Imagery: A survey", *Trans. PAMI*, vol. 37, no. 7, pp. 1480–1500, 2015.

[201] H. Zhao, C. Gan, A. Rouditchenko, C. Vondrick, J. McDermott, and A. Torralba, "The Sound of Pixels", in *ECCV*, 2018.

[202] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. S. Torr, "Conditional Random Fields as Recurrent Neural Networks", in *ICCV*, 2015.

[203] C. L. Zitnick and P. Dollár, "Edge Boxes: Locating Object Proposals from Edges", in *ECCV*, 2014.

[204] C. L. Zitnick, N. Jojic, and S. B. Kang, "Consistent segmentation for optical flow estimation", in *ICCV*, 2005.